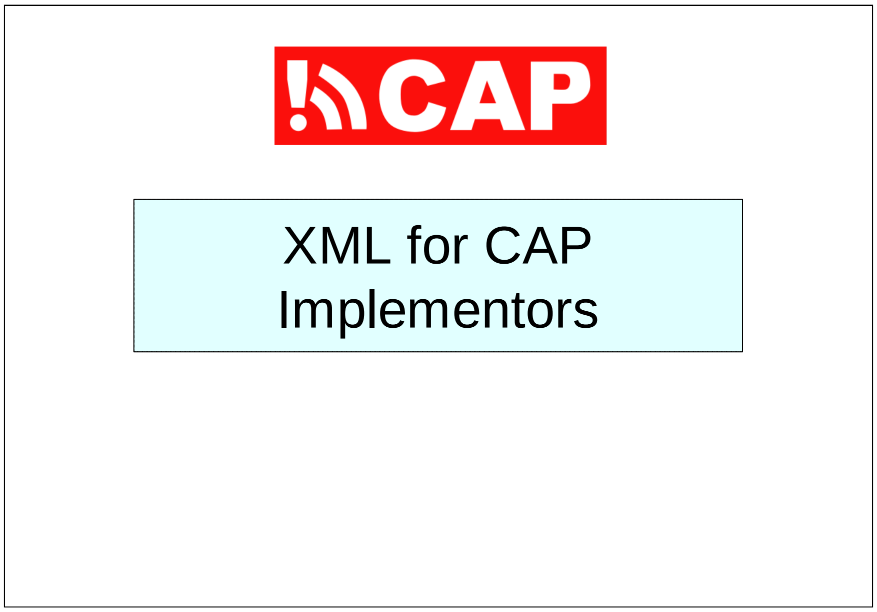


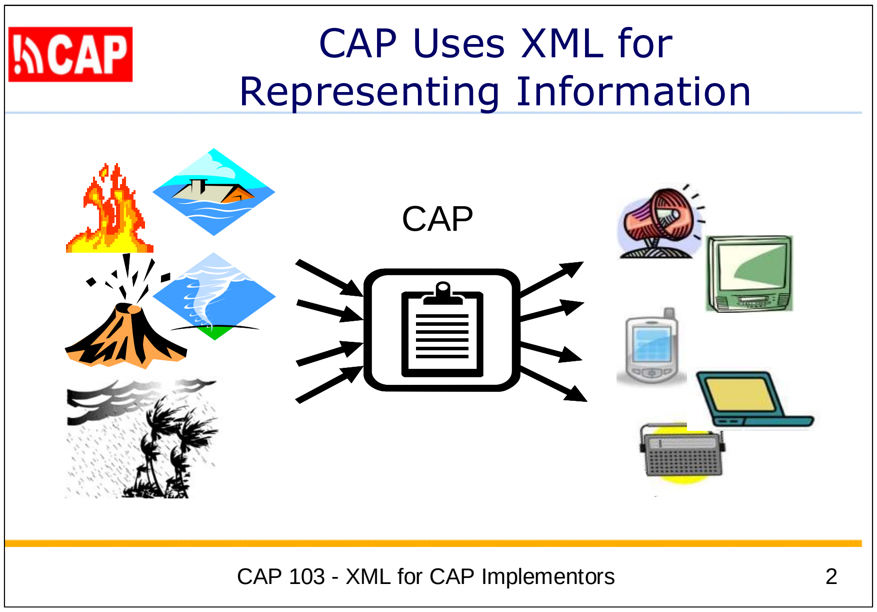
!CAP

XML for CAP Implementors

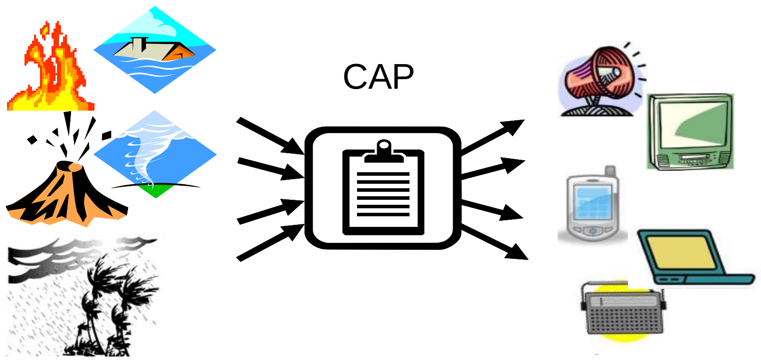
The title of this a presentation is “XML for CAP Implementors”.

This presentation is designed to be part of a series of training sessions that cover various aspects pertaining to CAP-enabled alerting systems.

The recommended pre-requisite for this session is “Introducing CAP” (CAP-101) .



CAP Uses XML for Representing Information



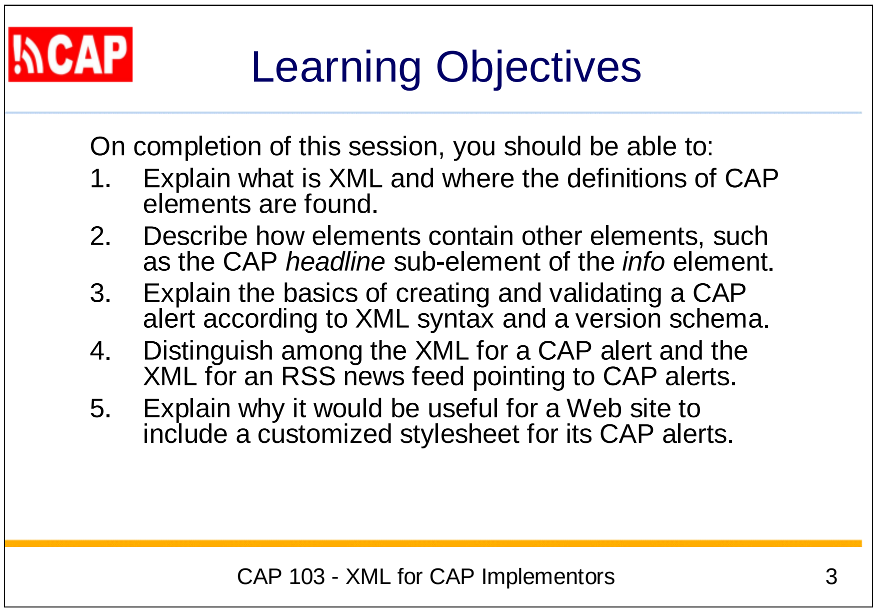
CAP 103 - XML for CAP Implementors 2

This diagram depicts CAP as a standard form, providing a common way of representing certain information on just about any kind of hazard threat or event.

The idea is to get some of that crucial information in the same format, so it is easier for people to process what they need to know.

Extensible Markup Language (XML) is the standard mechanism used to represent the information in CAP alerts and in CAP news feeds . So, some familiarity with XML is very important for a CAP implementor.

Let’s look at what is in the training session specifically.



Learning Objectives

On completion of this session, you should be able to:

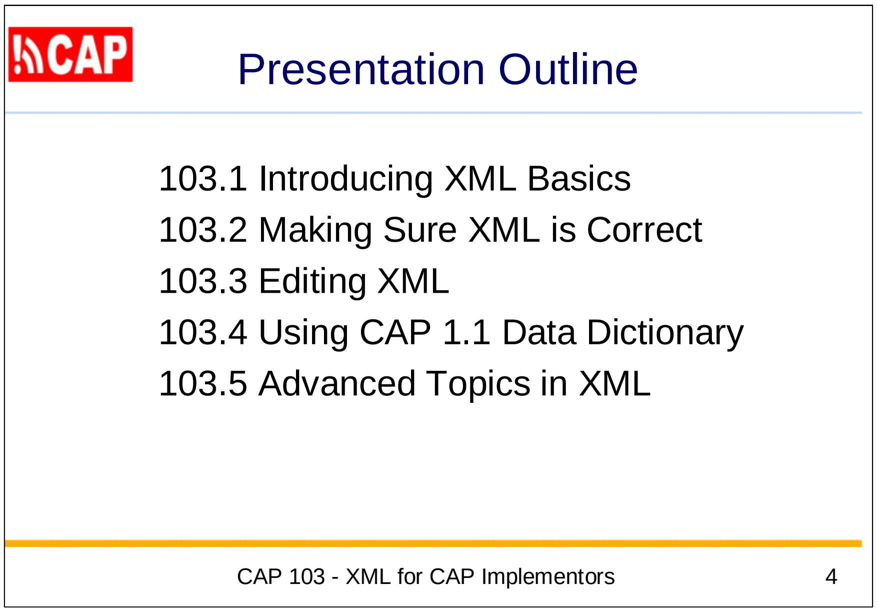
1. Explain what is XML and where the definitions of CAP elements are found.
2. Describe how elements contain other elements, such as the CAP *headline* sub-element of the *info* element.
3. Explain the basics of creating and validating a CAP alert according to XML syntax and a version schema.
4. Distinguish among the XML for a CAP alert and the XML for an RSS news feed pointing to CAP alerts.
5. Explain why it would be useful for a Web site to include a customized stylesheet for its CAP alerts.

CAP 103 - XML for CAP Implementors 3

Here are the Learning Objectives for this session.

On completion of this session, you should be able to:

1. Explain what is XML and where the definitions of CAP elements are found.
2. Describe how elements contain other elements, such as the CAP *headline* sub-element of the *info* element.
3. Explain the basics of creating and validating a CAP alert according to XML syntax and a specific version schema.
4. Distinguish among the XML for a CAP alert and the XML for an RSS news feed pointing to CAP alerts.
5. Explain why it would be useful for a Web site to include a customized stylesheet for its CAP alerts.



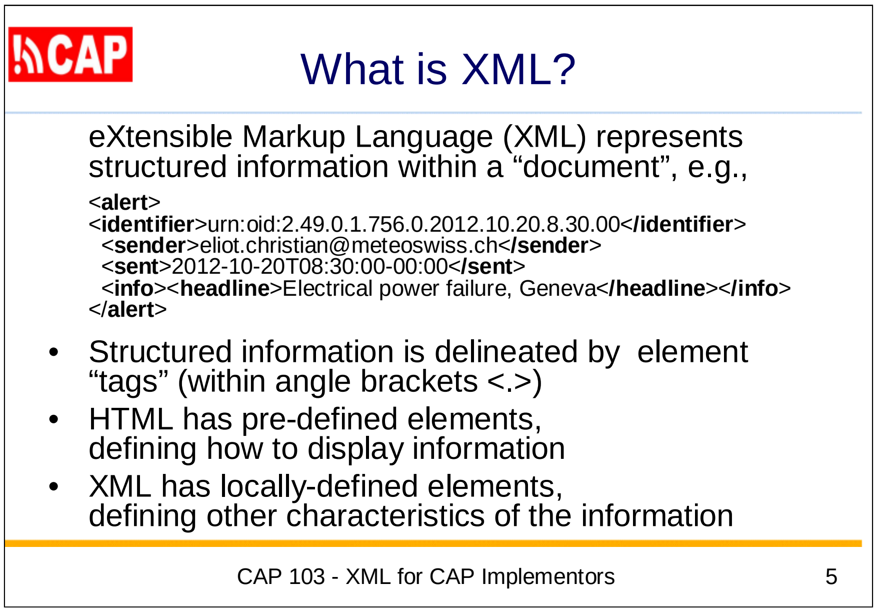
Presentation Outline

103.1 Introducing XML Basics
103.2 Making Sure XML is Correct
103.3 Editing XML
103.4 Using CAP 1.1 Data Dictionary
103.5 Advanced Topics in XML

CAP 103 - XML for CAP Implementors 4

Here is an outline of the major topics in this presentation.

The first topic is titled: Introducing XML Basics



What is XML?

eXtensible Markup Language (XML) represents structured information within a "document", e.g.,

```
<alert>
<identifier>urn:oid:2.49.0.1.756.0.2012.10.20.8.30.00</identifier>
  <sender>eliot.christian@meteoswiss.ch</sender>
  <sent>2012-10-20T08:30:00-00:00</sent>
  <info><headline>Electrical power failure, Geneva</headline></info>
</alert>
```

- Structured information is delineated by element "tags" (within angle brackets <.>)
- HTML has pre-defined elements, defining how to display information
- XML has locally-defined elements, defining other characteristics of the information

CAP 103 - XML for CAP Implementors 5

eXtensible Markup Language (XML) provides mechanisms to represent structured information within a container called a "document". Structured parts of the information are delineated by "tags" that are inserted as "markup" tokens using angle brackets.

HTML is also a markup language, but in HTML the tags are pre-defined in a generic HTML schema. HTML tags are mostly concerned with how to display the information (e.g., <b> for bold).

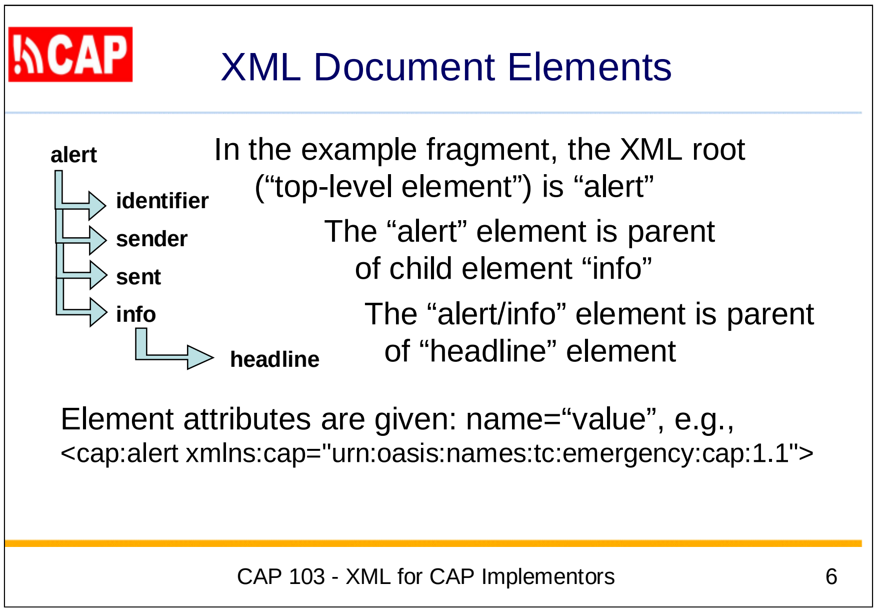
In XML, the tags are mostly concerned with delineating among different pieces of information. XML tags are locally-defined in a shema peculiar to that set of information. It has been said that HTML focuses on how the pieces of information should look while XML focuses on what the pieces of information are .

Here we see an example of a fragment of a CAP message in XML:

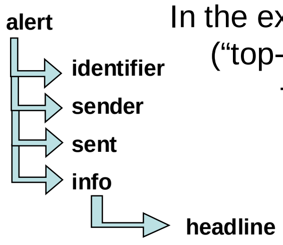
```
<alert>
 <identifier>urn:oid:2.49.0.1.756.0.2012.10.20.8.30.00</identifier>
  <sender>eliot.christian@meteoswiss.ch</sender>
  <sent>2012-10-20T08:30:00-00:00</sent>
  <info>
     <headline>Electrical power failure, Geneva</headline>
   </info>
</alert>
```

The tags in this example (alert, identifier, sender, sent, info, headline) are defined in the CAP schema.

Notice that each element begins with its "start tag" and ends with its "end tag". As in HTML, the value of the end tag is always the same as the start tag except that it begins with a forward slash. For instance, the "alert" element in this example is the entire fragment.



XML Document Elements



In the example fragment, the XML root ("top-level element") is "alert"

The "alert" element is parent of child element "info"

The "alert/info" element is parent of "headline" element

Element attributes are given: name="value", e.g.,
<cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.1">

CAP 103 - XML for CAP Implementors 6

"Elements" are the major pieces of an XML document. The XML document is said to have a "tree structure", with its "top-level element" also called its "root" element. Every XML document has a root element, and in CAP that element has the tag name "alert".

Each element of the XML document can contain one or more other elements. This feature gives the document a branching structure, looking at elements as branching nodes.

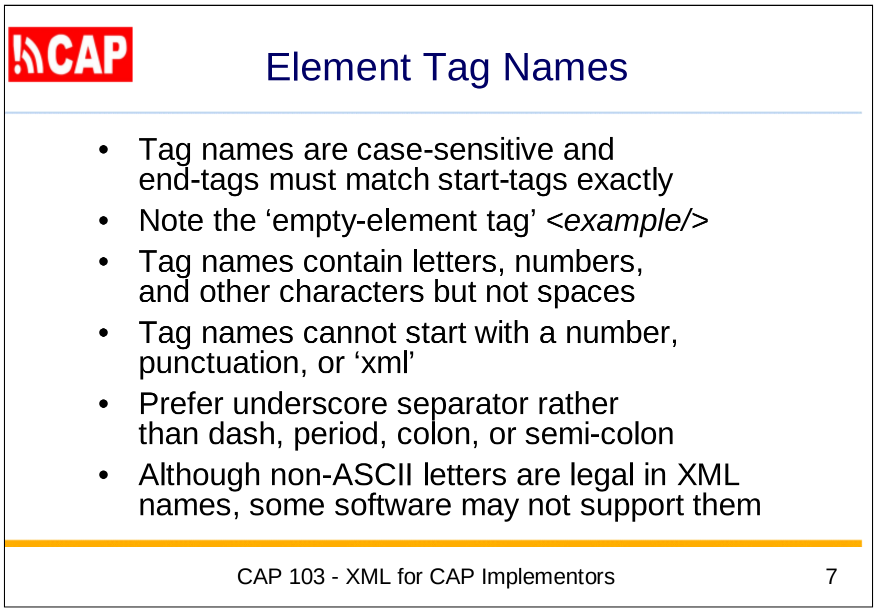
A node that contains other nodes is said to be the "parent" node; nodes contained within it are called its "child" nodes. Child nodes on the same level are called "siblings" of each other.

It is important to note that XML Elements **must be properly nested**. (HTML in practice often tolerates improper nesting of tags.)

In addition to containing other elements, each elements can have text content and it can have "attributes" as well.

Attributes are given in name=value pairs and the attribute value must always have quotes around it. The namespace attribute "**xmlns**" is required in a CAP document, as an attribute of the root element as shown here.

In comparison to elements, attributes are more difficult to read and maintain, they cannot contain multiple values, and they are not as easily expandable later. Accordingly, the CAP schema avoids attributes in favor of elements.



Element Tag Names

- Tag names are case-sensitive and end-tags must match start-tags exactly
- Note the 'empty-element tag' <*example/*>
- Tag names contain letters, numbers, and other characters but not spaces
- Tag names cannot start with a number, punctuation, or 'xml'
- Prefer underscore separator rather than dash, period, colon, or semi-colon
- Although non-ASCII letters are legal in XML names, some software may not support them

CAP 103 - XML for CAP Implementors 7

XML tag names are case-sensitive, and the start and end tags must match exactly, including the case of the letters in the tag name.

Here, I want to note the special case of an empty-element tag. If you have an end-tag immediately following its start-tag, then the element has a value of "empty". This can also be coded using just a single tag, where the forward slash is at the right end of the tag.

There are several rules for tag names of XML elements:

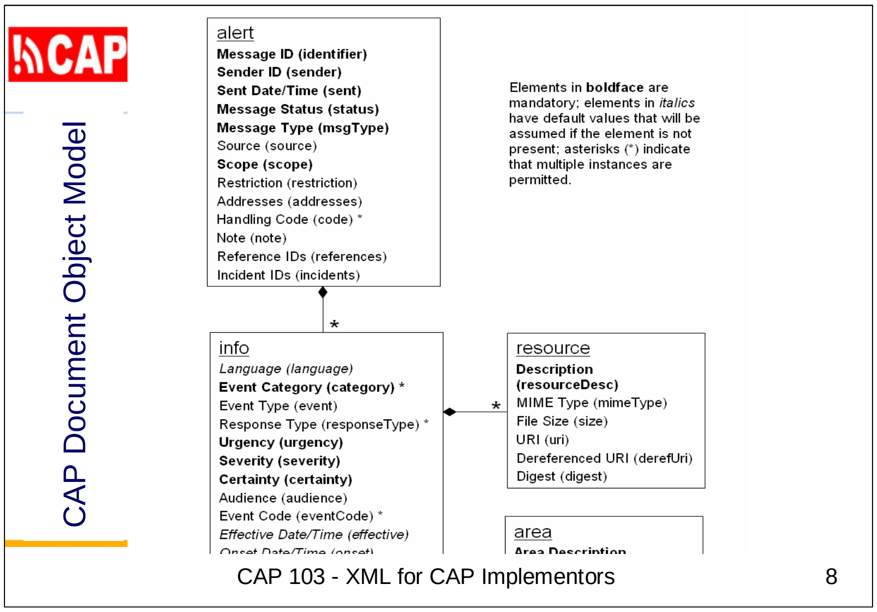
• Names can contain letters, numbers, and other characters but not spaces

• Names cannot start with a number, a punctuation character, or the letters xml (in any case)

• Names in XML should use underscore for a separator rather than a dash, period, colon, or semi-colon.

• Although non-ASCII letters legal in XML names, some software may not support them.

In the CAP schema, element names follow the convention known as "lower camel case". In "camel case", multiple words of a name are just concatenated in lower case except that a capital letter denotes each new word (and so sticks up like the hump of a camel). Then, depending on whether the first letter of the whole name is capitalized, we have "upper camel case" or "lower camel case".

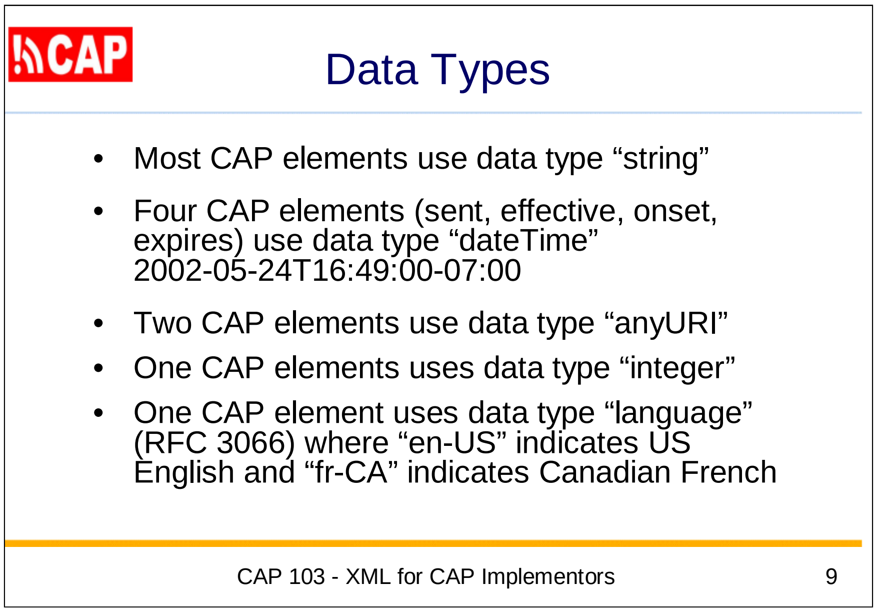


Here we see all of the CAP element names.

This is the CAP Document Object Model, as it is shown in the specification for CAP version 1.1.

You can see that all elements are shown, including their nested relationships. Also, all mandatory elements are shown in **bold**.

We will be looking at these elements in detail.



Data Types

- Most CAP elements use data type “string”
- Four CAP elements (sent, effective, onset, expires) use data type “dateTime” 2002-05-24T16:49:00-07:00
- Two CAP elements use data type “anyURI”
- One CAP elements uses data type “integer”
- One CAP element uses data type “language” (RFC 3066) where “en-US” indicates US English and “fr-CA” indicates Canadian French

CAP 103 - XML for CAP Implementors 9

In the CAP schema, most of the elements have the data type “string” which means they can contain any text.

Four elements in the CAP schema have the data type “dateTime”: **sent**, **effective**, **onset**, and **expires**.

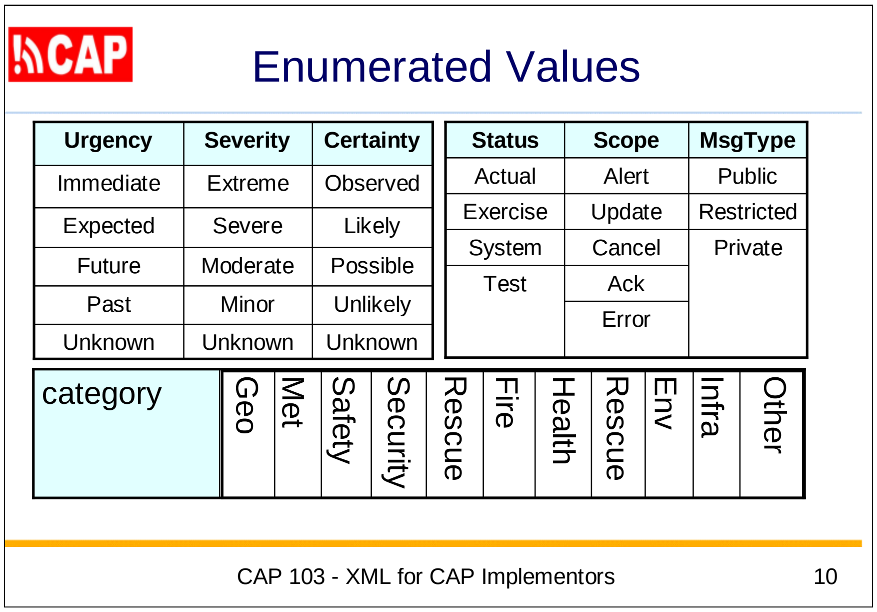
Here is an example of the dateTime format:
2002-05-24T16:49:00-07:00

which is 24 May 2002 at 16: 49 Pacific Daylight time, an offset of seven hours from UTC (Universal Time Coordinated, which you might have known as Greenwich Mean Time or “Z” / “Zulu”.

Two elements have the data type “anyURI”, which usually contains a URL.

One element has the data type “integer”.

One other element has the data type “language”, which is the data type commonly used on the Internet.



Enumerated Values

Urgency	Severity	Certainty
Immediate	Extreme	Observed
Expected	Severe	Likely
Future	Moderate	Possible
Past	Minor	Unlikely
Unknown	Unknown	Unknown

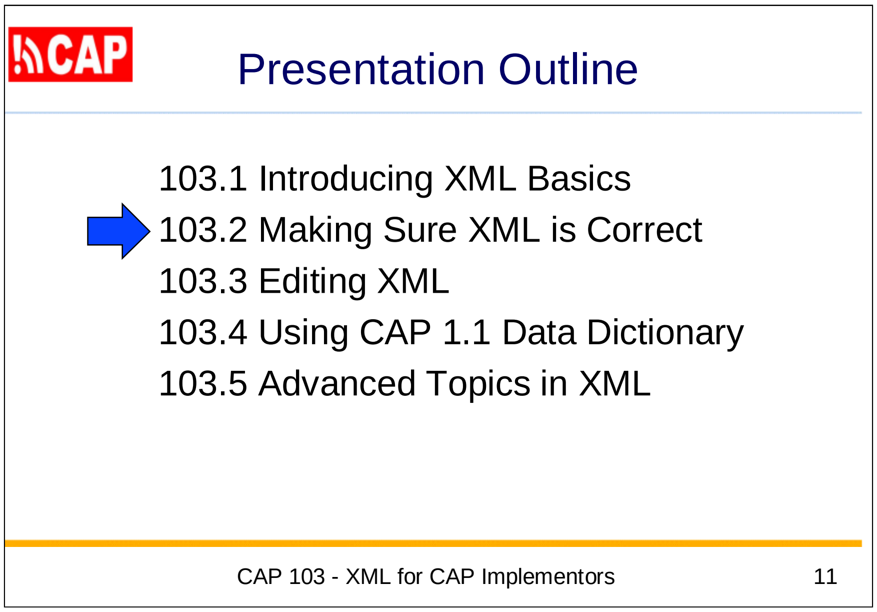
Status	Scope	MsgType
Actual	Alert	Public
Exercise	Update	Restricted
System	Cancel	Private
Test	Ack	
	Error	

category	Geo	Met	Safety	Security	Rescue	Fire	Health	Rescue	Env	Infra	Other

CAP 103 - XML for CAP Implementors 10

Many of the elements in a CAP alert are constrained to one or more choices from among a list of “enumerated values”. These values must be entered exactly as given in the schema.

When a CAP alert is edited with a Web form, the values would be selected through a “pull-down” list or “check boxes”.

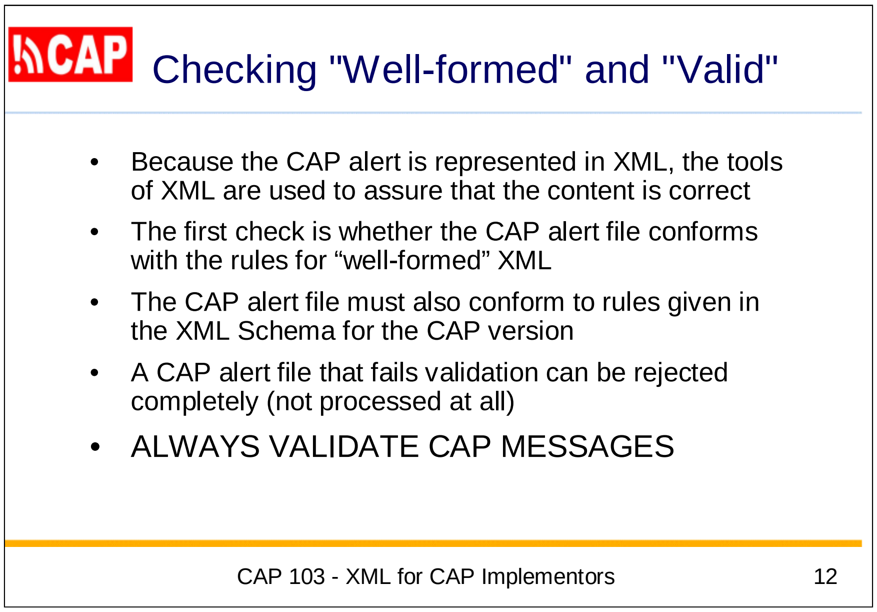


Presentation Outline

CAP 103 - XML for CAP Implementors 11

Again, here is the presentation outline.

The next topic is titled:
Making Sure XML is Correct



Checking "Well-formed" and "Valid"

- Because the CAP alert is represented in XML, the tools of XML are used to assure that the content is correct
- The first check is whether the CAP alert file conforms with the rules for “well-formed” XML
- The CAP alert file must also conform to rules given in the XML Schema for the CAP version
- A CAP alert file that fails validation can be rejected completely (not processed at all)
- ALWAYS VALIDATE CAP MESSAGES

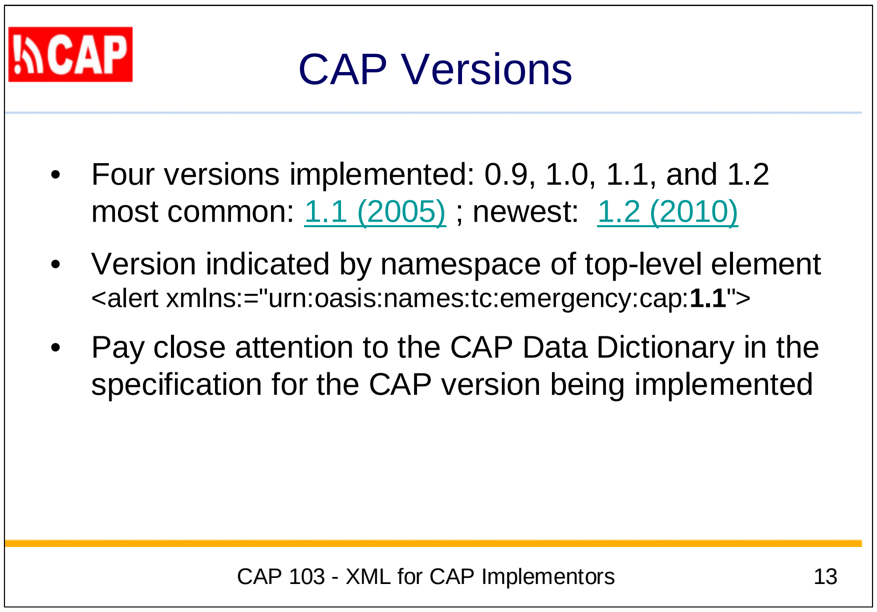
CAP 103 - XML for CAP Implementors 12

To check a CAP alert represented in XML, we need to use the tools of XML to assure that the content is correct.

The first check is whether the CAP alert file conforms to the rules for “well-formed” XML. That is: the XML document has a root element, all elements have matched start and end tags, nested elements are properly nested, and any attribute values are quoted.

The CAP alert file must also conform to rules given in the XML Schema for the CAP version. That XML Schema is found in the official specification of each CAP version, as published by OASIS and ITU.

A CAP alert file that fails validation can be rejected completely (not processed at all). Failing validation could result in an alert failing to be sent to many people who should have been alerted, so it is **very important** to always validate CAP messages!



CAP Versions

- Four versions implemented: 0.9, 1.0, 1.1, and 1.2 most common: 1.1 (2005) ; newest: 1.2 (2010)
- Version indicated by namespace of top-level element <alert xmlns:="urn:oasis:names:tc:emergency:cap:**1.1**">
- Pay close attention to the CAP Data Dictionary in the specification for the CAP version being implemented

CAP 103 - XML for CAP Implementors 13

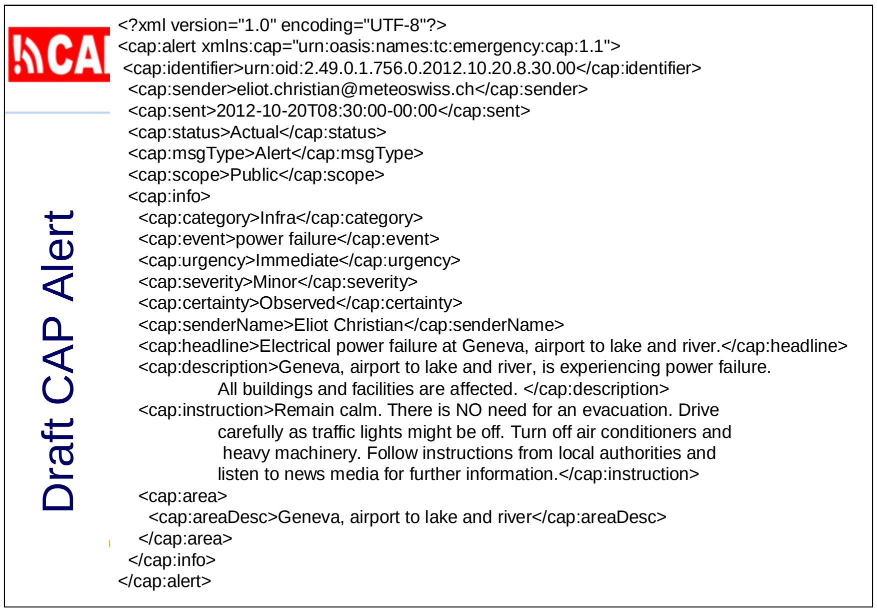
Four versions of CAP have been widely implemented: 0.9, 1.0, 1.1, and 1.2. Today, the most common version is version 1.1 (2005) although the newest version is 1.2 (2010).

Most CAP servers accept version 1.1; many accept 1.1. and 1.2.

The version of CAP being used is given in the namespace attribute of the top-level element (“alert”) in the CAP XML. For example,
this is what you would find in the XML for an alert in CAP v. 1.1 <alert xmlns="urn:oasis:names:tc:emergency:cap:1.1">

Pay close attention to the CAP Data Dictionary provided in the official specification for the CAP version that you are implementing.

In addition to conformance with a CAP version, an alerting authority may be required to also make its alert conform with a CAP Profile.
A Profile puts additional constraints on a CAP alert, but first the
CAP alert MUST be valid to a CAP version.



Draft CAP Alert

```
<?xml version="1.0" encoding="UTF-8"?>
<cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.1">
 <cap:identifier>urn:oid:2.49.0.1.756.0.2012.10.20.8.30.00</cap:identifier>
  <cap:sender>eliot.christian@meteoswiss.ch</cap:sender>
  <cap:sent>2012-10-20T08:30:00-00:00</cap:sent>
  <cap:status>Actual</cap:status>
  <cap:msgType>Alert</cap:msgType>
  <cap:scope>Public</cap:scope>
  <cap:info>
    <cap:category>Infra</cap:category>
    <cap:event>power failure</cap:event>
    <cap:urgency>Immediate</cap:urgency>
    <cap:severity>Minor</cap:severity>
    <cap:certainty>Observed</cap:certainty>
    <cap:senderName>Eliot Christian</cap:senderName>
    <cap:headline>Electrical power failure at Geneva, airport to lake and river.</cap:headline>
    <cap:description>Geneva, airport to lake and river, is experiencing power failure.
                All buildings and facilities are affected. </cap:description>
    <cap:instruction>Remain calm. There is NO need for an evacuation. Drive
                carefully as traffic lights might be off. Turn off air conditioners and
                 heavy machinery. Follow instructions from local authorities and
                listen to news media for further information.</cap:instruction>
    <cap:area>
      <cap:areaDesc>Geneva, airport to lake and river</cap:areaDesc>
    </cap:area>
  </cap:info>
</cap:alert>
```

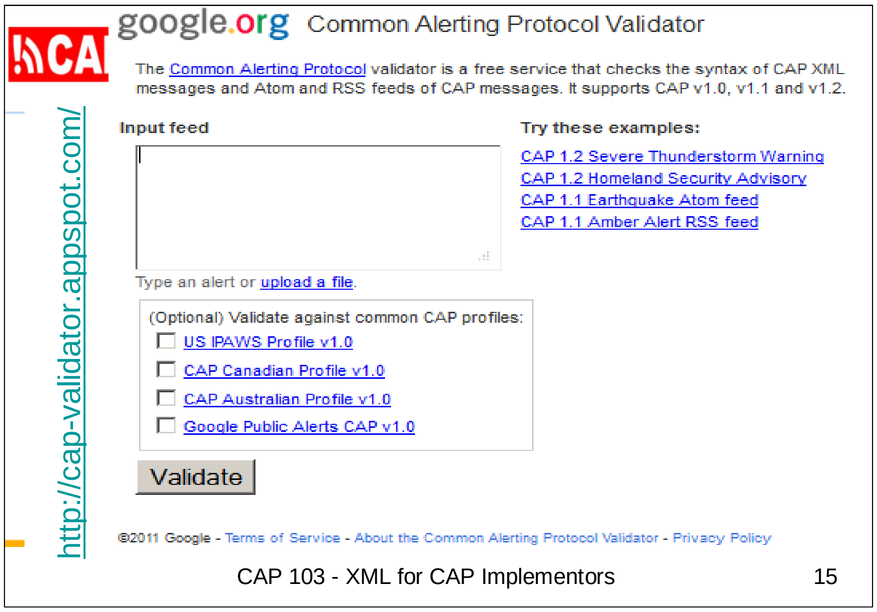
In a later session (CAP-105), you can work through exactly how to select appropriate values for a particular CAP alert. Here I am simply showing a complete draft CAP alert so you can see its XML format.

XML can be entered “freeform” using any text editor. We will have some more to note later about certain editing cautions.

Now we should look at the first line of the XML file. This is known as the XML Processing Instruction. It is identified by the **?xml** where you would expect a tag if it were an element.

You see here the processing instruction has two attributes: version and encoding. For a CAP alert, you should enter this line exactly as shown here.

Once you are finished editing the file, you should save it with a “.xml” file extension. Also, be sure to use “save as” and select “UTF-8” encoding, as will be explained later.



Common Alerting Protocol Validator

The Common Alerting Protocol validator is a free service that checks the syntax of CAP XML messages and Atom and RSS feeds of CAP messages. It supports CAP v1.0, v1.1 and v1.2.

Input feed

Type an alert or upload a file.

(Optional) Validate against common CAP profiles:
- ☐ US IPAWS Profile v1.0
- ☐ CAP Canadian Profile v1.0
- ☐ CAP Australian Profile v1.0
- ☐ Google Public Alerts CAP v1.0

Try these examples:
- CAP 1.2 Severe Thunderstorm Warning
- CAP 1.2 Homeland Security Advisory
- CAP 1.1 Earthquake Atom feed
- CAP 1.1 Amber Alert RSS feed

©2011 Google - Terms of Service - About the Common Alerting Protocol Validator - Privacy Policy

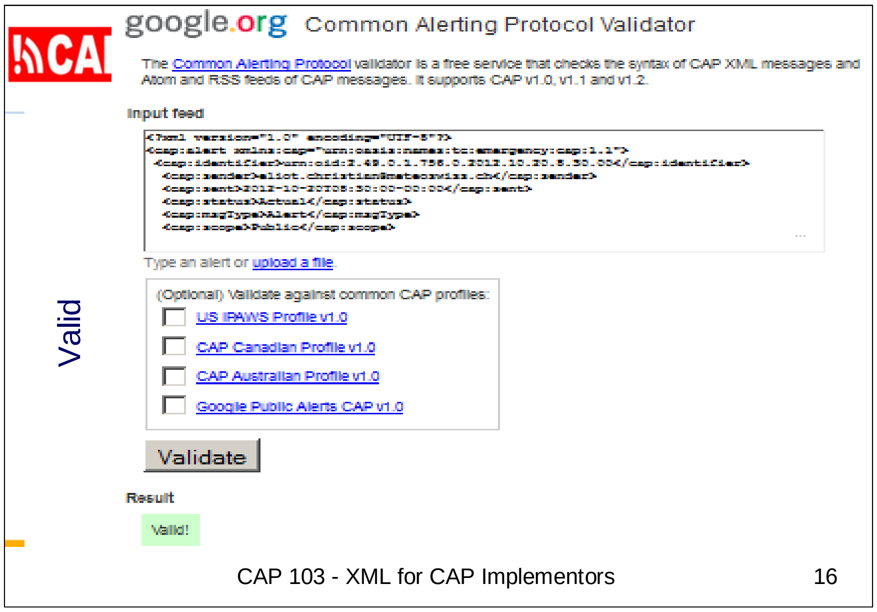
CAP 103 - XML for CAP Implementors 15

The XML validation function is included in all programming languages commonly used with Web sites (Java, PHP, Perl,
Visual Basic, etc.) I'm not going to show how to invoke XML validation as a programmer. Suffice it to say that you will need
to point to the appropriate CAP schema (by version) as well as pointing to the CAP alert file.

What we will use now is an online CAP validator from Google.
This validator supports CAP versions 1.0, 1.1, and 1.2. It prompts you to either paste the CAP XML directly into the text box or to use the upload link (which will then prompt you to point to the CAP alert file on your local system).

I will now run this validator using the CAP alert we just created.

So, we see that this CAP alert is well-formed XML and it is valid as CAP version 1.1.



google.org Common Alerting Protocol Validator

The Common Alerting Protocol validator is a free service that checks the syntax of CAP XML messages and Atom and RSS feeds of CAP messages. It supports CAP v1.0, v1.1 and v1.2.

Input feed

```
<?xml version="1.0" encoding="UTF-8"?>
<cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.1">
 <cap:identifier>urn:oid:2.49.0.1.756.0.2012.10.20.8.30.00</cap:identifier>
  <cap:sender>eliot.christian@meteoswiss.ch</cap:sender>
  <cap:sent>2012-10-20T08:30:00-00:00</cap:sent>
  <cap:status>Actual</cap:status>
  <cap:msgType>Alert</cap:msgType>
  <cap:scope>Public</cap:scope>
```

Type an alert or upload a file.

(Optional) Validate against common CAP profiles:

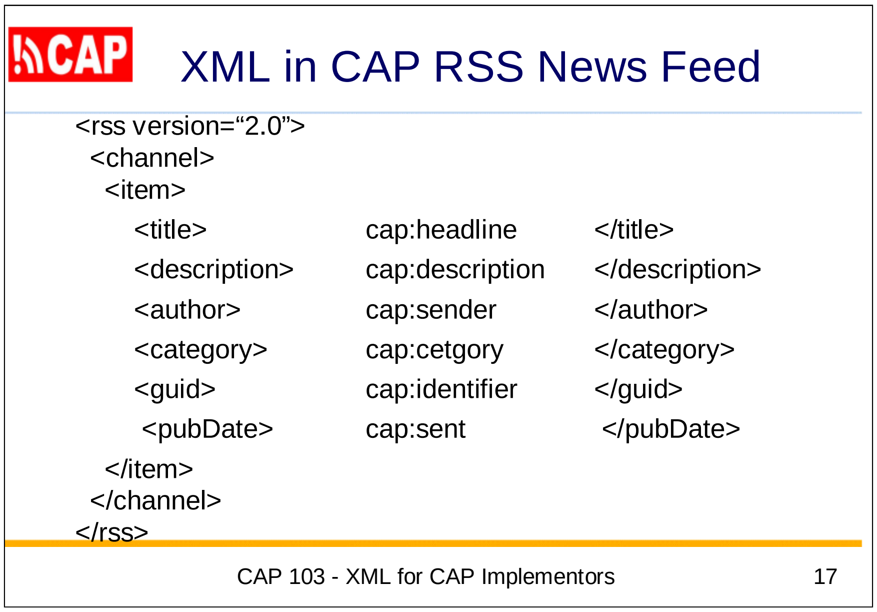
- ☐ US IPAWS Profile v1.0
- ☐ CAP Canadian Profile v1.0
- ☐ CAP Australian Profile v1.0
- ☐ Google Public Alerts CAP v1.0

Result

Valid!

CAP 103 - XML for CAP Implementors 16

In case we don't have live Internet for this presentation, here is a screen shot showing the result from running the validator--the CAP alert is **valid** (green box).



XML in CAP RSS News Feed

```
<rss version=“2.0”>
  <channel>
    <item>
        <title>           cap:headline        </title>
        <description>     cap:description     </description>
        <author>          cap:sender          </author>
        <category>        cap:cetgory         </category>
        <guid>            cap:identifier      </guid>
         <pubDate>        cap:sent             </pubDate>
    </item>
  </channel>
</rss>
```

CAP 103 - XML for CAP Implementors 17

Now let’s look at XML in CAP RSS News Feed.

In an RSS feed, the top level element is “rss” and it has a mandatory version attribute.

Then, there is a single sub-element: “channel”.

The channel may contain any number of <item> sub-elements.

We can regard a source of CAP alerts as an RSS channel and each RSS item corresponds to one CAP alert.

For the RSS item/title, we can use cap:headline.

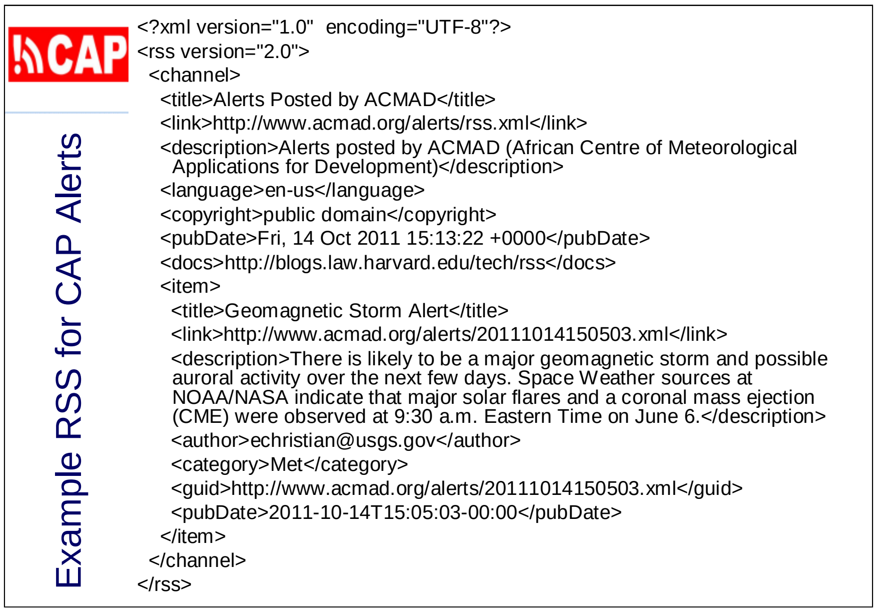
For the RSS item/description, use cap:description

For item/author, use cap:sender

For item/category, use cap:category

For the item/guid, use cap:identifier

And, for the item/pubDate, use cap:sent.

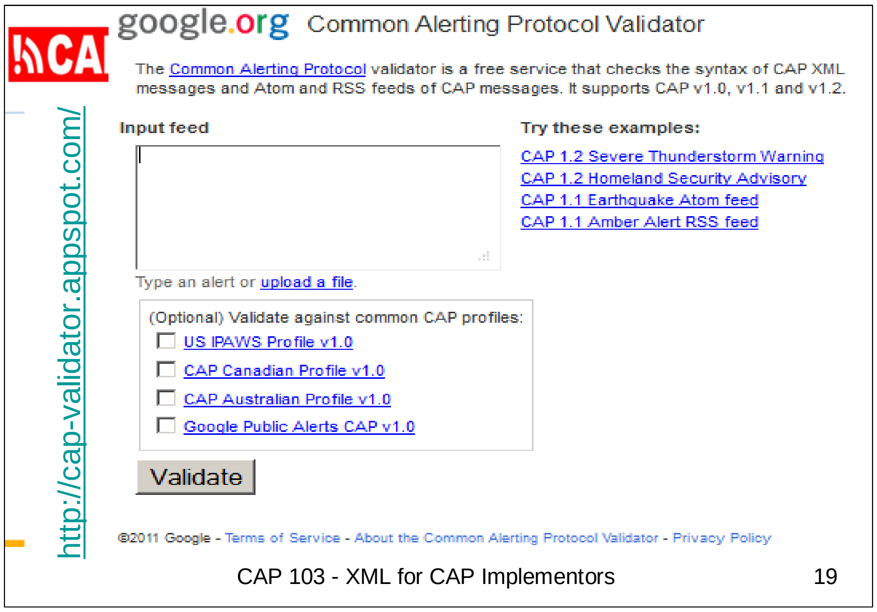


Example RSS for CAP Alerts

```
<?xml version="1.0"  encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title>Alerts Posted by ACMAD</title>
    <link>http://www.acmad.org/alerts/rss.xml</link>
    <description>Alerts posted by ACMAD (African Centre of Meteorological
      Applications for Development)</description>
    <language>en-us</language>
    <copyright>public domain</copyright>
    <pubDate>Fri, 14 Oct 2011 15:13:22 +0000</pubDate>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <item>
      <title>Geomagnetic Storm Alert</title>
      <link>http://www.acmad.org/alerts/20111014150503.xml</link>
      <description>There is likely to be a major geomagnetic storm and possible
      auroral activity over the next few days. Space Weather sources at
      NOAA/NASA indicate that major solar flares and a coronal mass ejection
      (CME) were observed at 9:30 a.m. Eastern Time on June 6.</description>
      <author>echristian@usgs.gov</author>
      <category>Met</category>
      <guid>http://www.acmad.org/alerts/20111014150503.xml</guid>
      <pubDate>2011-10-14T15:05:03-00:00</pubDate>
    </item>
  </channel>
</rss>
```

Now here is an example of XML as it could be used with a CAP news feed. In session CAP-106 we look at exactly how to create such an XML file.

Here I am just showing a single item. Of course, a real news feed would have multiple items, corresponding to multiple alerts.


google.org Common Alerting Protocol Validator
http://cap-validator.appspot.com/

The Common Alerting Protocol validator is a free service that checks the syntax of CAP XML messages and Atom and RSS feeds of CAP messages. It supports CAP v1.0, v1.1 and v1.2.

Input feed

Type an alert or upload a file.

(Optional) Validate against common CAP profiles:
- US IPAWS Profile v1.0
- CAP Canadian Profile v1.0
- CAP Australian Profile v1.0
- Google Public Alerts CAP v1.0

Validate

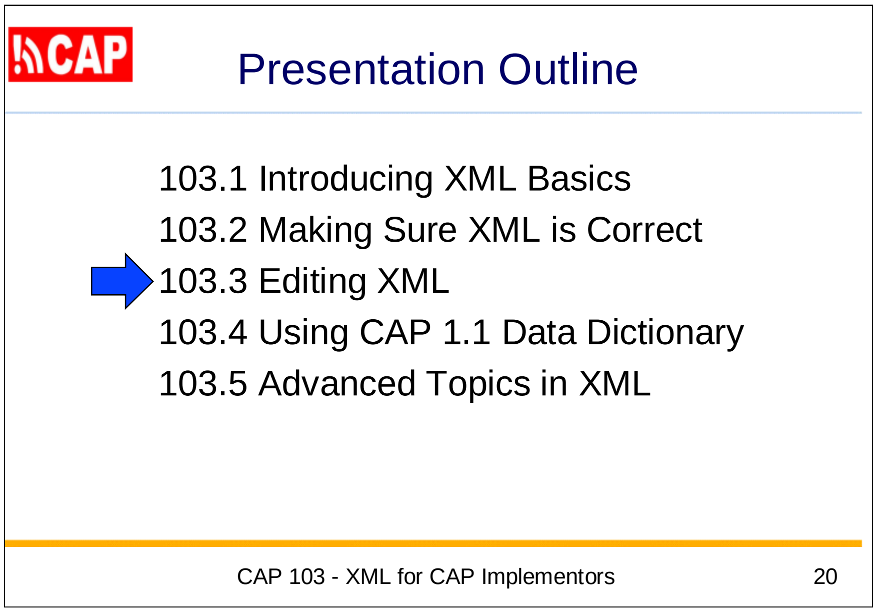
Try these examples:
- CAP 1.2 Severe Thunderstorm Warning
- CAP 1.2 Homeland Security Advisory
- CAP 1.1 Earthquake Atom feed
- CAP 1.1 Amber Alert RSS feed

©2011 Google - Terms of Service - About the Common Alerting Protocol Validator - Privacy Policy

CAP 103 - XML for CAP Implementors 19

Again we will use the online validator tool from Google. This time we will validate the CAP RSS news feed.

The feed itself doesn’t receive any error messages, but the validation process also tries to validate each of the CAP alerts. In this draft feed, the alerts URLs are bogus, so that part fails.



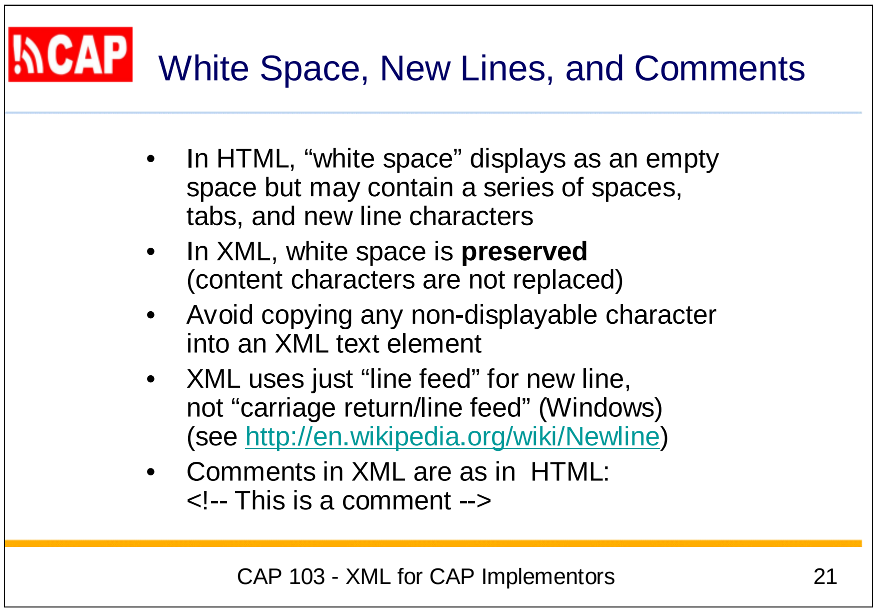
Presentation Outline

103.1 Introducing XML Basics
103.2 Making Sure XML is Correct
103.3 Editing XML
103.4 Using CAP 1.1 Data Dictionary
103.5 Advanced Topics in XML

CAP 103 - XML for CAP Implementors 20

Going back to the presentation outline, we see the next topic is titled:

Editing XML



White Space, New Lines, and Comments

- In HTML, “white space” displays as an empty space but may contain a series of spaces, tabs, and new line characters
- In XML, white space is **preserved** (content characters are not replaced)
- Avoid copying any non-displayable character into an XML text element
- XML uses just “line feed” for new line, not “carriage return/line feed” (Windows) (see http://en.wikipedia.org/wiki/Newline)
- Comments in XML are as in HTML: <!-- This is a comment -->

CAP 103 - XML for CAP Implementors 21

The term “white space” has a special meaning in markup languages such as XML. In HTML, for instance, white space refers to certain characters that render as empty space on a display: not only a series of spaces but a tab, a line feed and a carriage return as well.

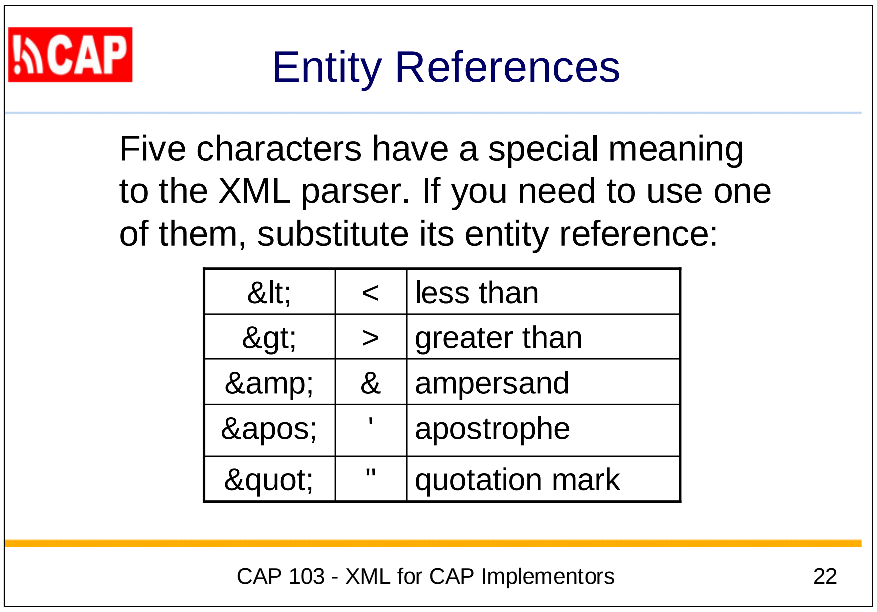
It is important to note that white space is **preserved** in XML. Unlike HTML, content characters are not replaced by a space character. You should also note that what appears to be a space as seen on your computer screen may be instead a *non-displayable character*.

Let’s say you are filling out a text element in CAP alert form and you copy some text from another document open on your desktop. If that text contains non-displayed characters, then you have just dropped hidden garbage into the CAP element. This can cause the XML to fail validation, or strange characters may be displayed when the alert is later viewed by someone else.

One way to minimize editing problems is to constantly display any white space characters in the text. You can also get freeware tools such as “Pure Text” that specifically strip non-displayed characters from the scratch pad so you can paste just clean text.

As you may know, computer applications have very confused ways of dealing with the “new line” commonly used to separate lines or close a section of text. Microsoft Windows codes a new line as a pair of characters: carriage return (CR) and line feed (LF), much like an old manual typewriters. Other applications use only a line feed character for a new line and that is the case for XML also. Further technical discussion on this is at http://en.wikipedia.org/wiki/Newline

We should note also that you can have comments in XML as you would in HTML: <!-- This is a comment --> **Note**: no comment is allowed to precede the XML declaration, which must be the first line.



Entity References

Five characters have a special meaning to the XML parser. If you need to use one of them, substitute its entity reference:

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

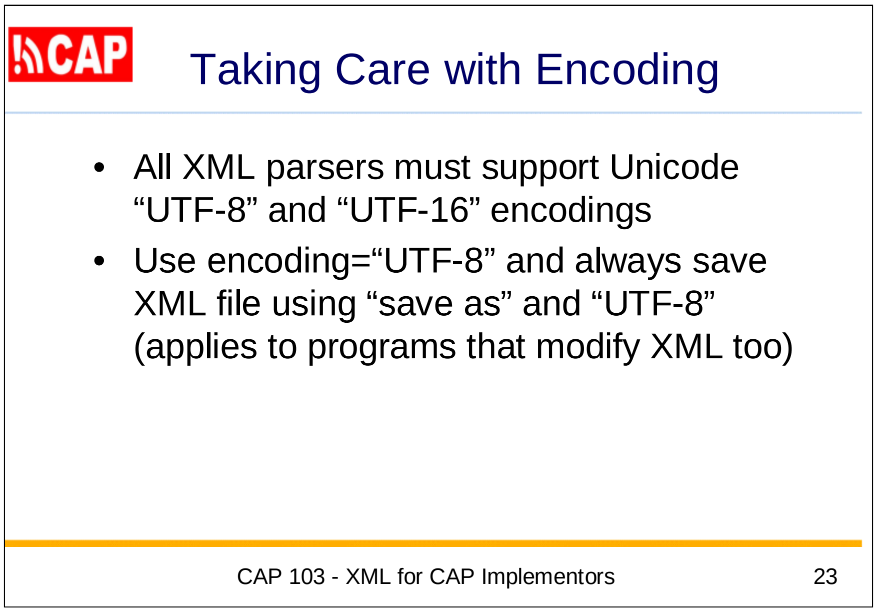
CAP 103 - XML for CAP Implementors 22

A few characters have a special meaning in XML and you need to take care to “escape” them with an entity reference instead.

As we’ve seen, XML tags are formed by using the left and right angle brackets (< and >), also known as the “less than” symbol and the “greater than” symbol.

The XML parser will give an error if the left angle bracket is used anywhere else in the uncommented part of an XML document.
If you do have text where the “less than” symbol must occur, you need to substitute for it the corresponding “entity reference”: **&lt;**

There are four other predefined entity references that usually substituted in an XML document to avoid confusion over their special meaning to the XML parser: **&gt;** for the “>” greater than symbol, **&amp;** for the “&” ampersand, **&apos; “‘”** for the apostrophe, and **&quot;** for the double quotation mark.



Taking Care with Encoding

- All XML parsers must support Unicode “UTF-8” and “UTF-16” encodings
- Use encoding=“UTF-8” and always save XML file using “save as” and “UTF-8” (applies to programs that modify XML too)

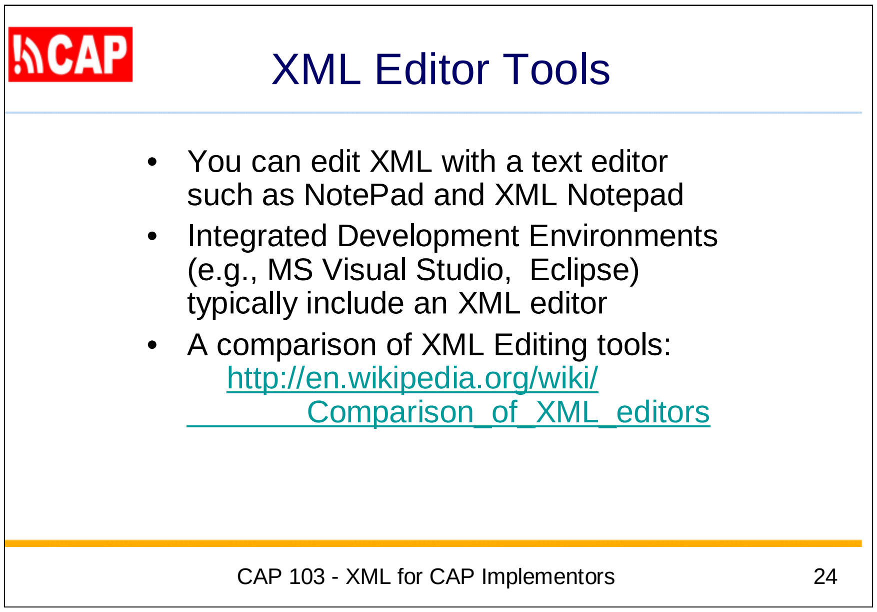
CAP 103 - XML for CAP Implementors 23

Encoding issues are common in dealing with XML. When attempting to load an XML file with a browser, for instance, you can get an invalid character error message.

I recommend that you always use UTF-8 (or UTF-16 if necessary for your natural language). All XML parsers are required to support Unicode “UTF-8” and “UTF-16”.

In the first line of your XML, always specify encoding=“UTF-8”. Always save the XML file using “**save as**” and “UTF-8”.

This applies also to program code that may modify any XML files.



XML Editor Tools

- You can edit XML with a text editor such as NotePad and XML Notepad
- Integrated Development Environments (e.g., MS Visual Studio, Eclipse) typically include an XML editor
- A comparison of XML Editing tools: http://en.wikipedia.org/wiki/Comparison_of_XML_editors

CAP 103 - XML for CAP Implementors 24

Most people dealing with CAP will not see the raw XML but will handle CAP alert information through a form. In the rare event they may need to edit the CAP XML directly, they can use a text editor such as regular NotePad or XML Notepad.

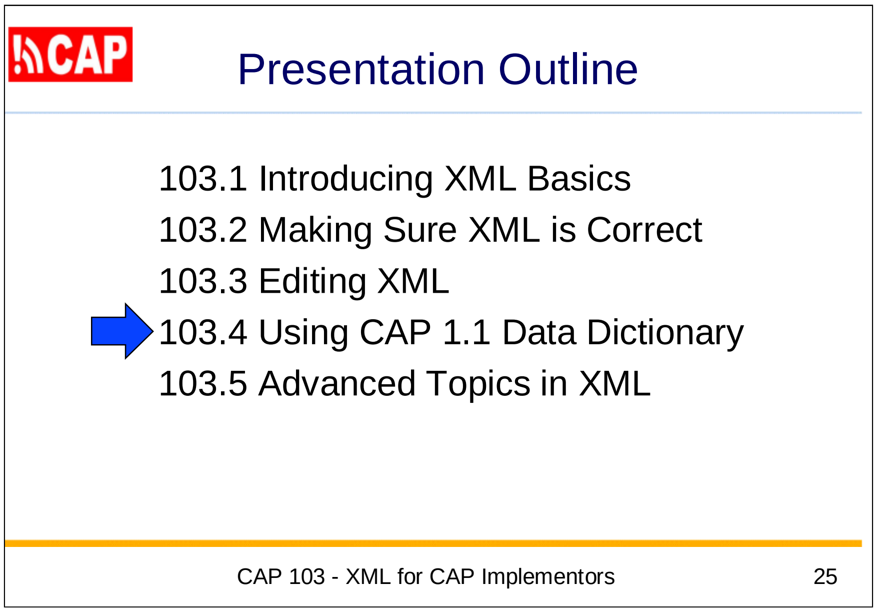
Developers and some others dealing with CAP may need to handle XML more often or more deeply. In those cases, it makes sense to have an editing tool specifically designed for XML.

An integrated development environment (including freeware such as Visual Studio or Eclipse) typically supports XML editing. But, that might be overkill for those with simple XML editing needs.

There are some freeware tools for XML editing. At minimum, an XML editing tool should provide syntax highlighting so as to differentiate element content from tags (as we saw in the Internet Explorer browser view of a CAP alert.)

There are also some quite pricey XML editing tools. One of the most popular is “XML Spy” but you can compare that and others online--see, for example:

http://en.wikipedia.org/wiki/Comparison_of_XML_editors



Presentation Outline

CAP 103 - XML for CAP Implementors 25

Our next topic concerns
Using the CAP 1.1 Data Dictionary

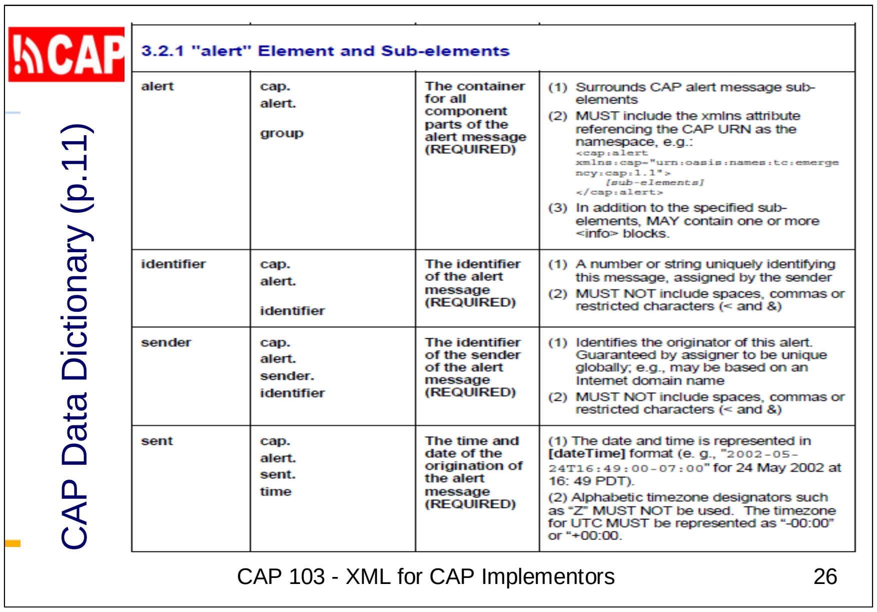
The Data Dictionary is a document for human readers. It provides an understandable description of the element of a CAP message, relationships among elements, and any value constraints on the elements.

If you were to look at the corresponding CAP XML Schema, you would find that the schema actually enforces many of the element relationships and value constraints. But, the schema has only part of the information given in the data dictionary, and an XML schema is designed for machine processing rather than human readers.

When a CAP message fails validation, the cause is usually a violation of XML Schema constraints. But, it is a good idea to also check the Data Dictionary whenever you see an unfamiliar value in a CAP element.

The data dictionary notes overall that CAP elements MAY have null values, unless null values are specifically prohibited for an element. (This is typical of XML—that a value can be empty or “null”.) Implementers MUST check for this condition wherever it might affect application performance.

The data dictionary starts on page 11 in the CAP 1.1 standard specification document. It is presented as a table with four columns.



CAP Data Dictionary (p.11)

3.2.1 "alert" Element and Sub-elements

alert	cap. alert. group	The container for all component parts of the alert message (REQUIRED)	(1) Surrounds CAP alert message sub-elements (2) MUST include the xmlns attribute referencing the CAP URN as the namespace, e.g.: `<cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.1">` *[sub-elements]* `</cap:alert>` (3) In addition to the specified sub-elements, MAY contain one or more <info> blocks.
identifier	cap. alert. identifier	The identifier of the alert message (REQUIRED)	(1) A number or string uniquely identifying this message, assigned by the sender (2) MUST NOT include spaces, commas or restricted characters (< and &)
sender	cap. alert. sender. identifier	The identifier of the sender of the alert message (REQUIRED)	(1) Identifies the originator of this alert. Guaranteed by assigner to be unique globally; e.g., may be based on an Internet domain name (2) MUST NOT include spaces, commas or restricted characters (< and &)
sent	cap. alert. sent. time	The time and date of the origination of the alert message (REQUIRED)	(1) The date and time is represented in **[dateTime]** format (e. g., "`2002-05-24T16:49:00-07:00`" for 24 May 2002 at 16: 49 PDT). (2) Alphabetic timezone designators such as "Z" MUST NOT be used. The timezone for UTC MUST be represented as "-00:00" or "+00:00.

CAP 103 - XML for CAP Implementors 26

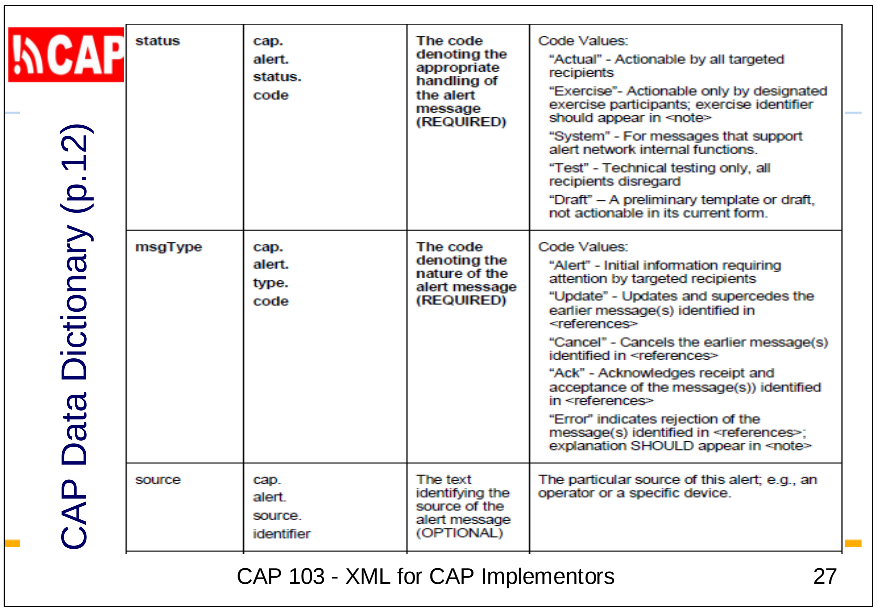
The first column has the name of the element.
Then there is given the formal construction of the element, following the style given in ISO 11179 (Metadata Registries).
The third column gives the definition of the element. In parentheses the dictionary notes whether this element is mandatory or optional. The fourth column provides any additional notes about usage of the element. In particular, this is where you would find the allowed values and constraints on the coding of the values in the element.

The first element described has the name “alert”. It is **REQUIRED** in every CAP message and is the container for all component parts of the alert message. Here we also see that **MUST** include the **xmlns** attribute. As we know, the alert elements **MAY** contain one or more “info” sub-elements.

The first five sub-elements are **REQUIRED** in every CAP alert message. First we see the element named **identifier** and note that this value must uniquely identify the message. The value of the identifier element **MUST NOT** include spaces, commas or restricted characters.

Next we see the element **sender**, which is guaranteed to be unique globally (and is often an e-mail address). The value of the sender element also MUST NOT include spaces, commas or restricted characters.

The third sub-element is the CAP alert **sent** time. The value of this element must be represented in [dateTime] format (e. g., "2002-05-24T16:49:00-07:00" for 24 May 2002 at 16: 49 Pacific Daylight Time (West coast of continental U.S.). It is also noted that timezone designators such as "Z" **MUST NOT** be used. Instead the zero offset from UTC is used, that is: "+00:00“ is the offset.



CAP Data Dictionary (p.12)

status	**cap. alert. status. code**	**The code denoting the appropriate handling of the alert message (REQUIRED)**	Code Values: "Actual" - Actionable by all targeted recipients "Exercise"- Actionable only by designated exercise participants; exercise identifier should appear in <note> "System" - For messages that support alert network internal functions. "Test" - Technical testing only, all recipients disregard "Draft" – A preliminary template or draft, not actionable in its current form.
msgType	**cap. alert. type. code**	**The code denoting the nature of the alert message (REQUIRED)**	Code Values: "Alert" - Initial information requiring attention by targeted recipients "Update" - Updates and supercedes the earlier message(s) identified in <references> "Cancel" - Cancels the earlier message(s) identified in <references> "Ack" - Acknowledges receipt and acceptance of the message(s)) identified in <references> "Error" indicates rejection of the message(s) identified in <references>; explanation SHOULD appear in <note>
source	cap. alert. source. identifier	The text identifying the source of the alert message (OPTIONAL)	The particular source of this alert; e.g., an operator or a specific device.

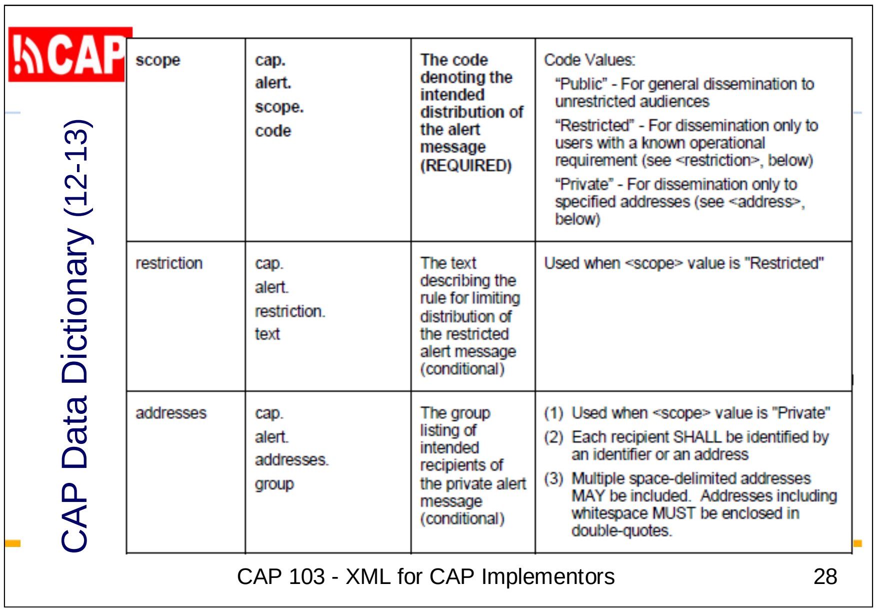
CAP 103 - XML for CAP Implementors 27

Here are data dictionary entries for the elements **status**"and **msgType**, two more of the first five required sub-elements.

The required element "status" denotes appropriate handling of the alert message. It has five allowed values: "Actual", "Exercise", "System", "Test", and "Draft".

The required element "msgType" also has five allowed values: "Alert", "Update", "Cancel", "Ack" (for "acknowledge receipt"), and "Error".

Next we see the optional element named "source" of type "text".



CAP Data Dictionary (12-13)

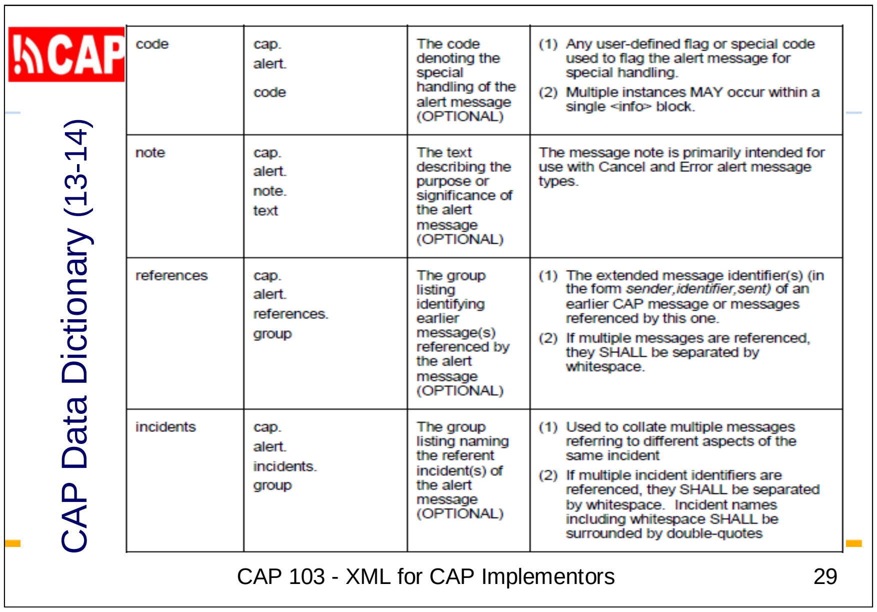
scope	**cap. alert. scope. code**	**The code denoting the intended distribution of the alert message (REQUIRED)**	Code Values: "Public" - For general dissemination to unrestricted audiences "Restricted" - For dissemination only to users with a known operational requirement (see <restriction>, below) "Private" - For dissemination only to specified addresses (see <address>, below)
restriction	cap. alert. restriction. text	The text describing the rule for limiting distribution of the restricted alert message (conditional)	Used when <scope> value is "Restricted"
addresses	cap. alert. addresses. group	The group listing of intended recipients of the private alert message (conditional)	(1) Used when <scope> value is "Private" (2) Each recipient SHALL be identified by an identifier or an address (3) Multiple space-delimited addresses MAY be included. Addresses including whitespace MUST be enclosed in double-quotes.

CAP 103 - XML for CAP Implementors 28

Now we see another **REQUIRED** element, **scope**. This element denotes the intended distribution. It has three allowed values: "Public", "Restricted", and "Private".

If the value of scope is "Restricted“, then there must be a value in the next element **restriction.** That is why the “optionality” is conditional rather than mandatory or optional.

If the value of scope is “Private“, then there must be a value in the next element **addresses.** Each addresses is separated from the next by a space.



CAP Data Dictionary (13-14)

code	cap. alert. code	The code denoting the special handling of the alert message (OPTIONAL)	(1) Any user-defined flag or special code used to flag the alert message for special handling. (2) Multiple instances MAY occur within a single <info> block.
note	cap. alert. note. text	The text describing the purpose or significance of the alert message (OPTIONAL)	The message note is primarily intended for use with Cancel and Error alert message types.
references	cap. alert. references. group	The group listing identifying earlier message(s) referenced by the alert message (OPTIONAL)	(1) The extended message identifier(s) (in the form *sender,identifier,sent)* of an earlier CAP message or messages referenced by this one. (2) If multiple messages are referenced, they SHALL be separated by whitespace.
incidents	cap. alert. incidents. group	The group listing naming the referent incident(s) of the alert message (OPTIONAL)	(1) Used to collate multiple messages referring to different aspects of the same incident (2) If multiple incident identifiers are referenced, they SHALL be separated by whitespace. Incident names including whitespace SHALL be surrounded by double-quotes

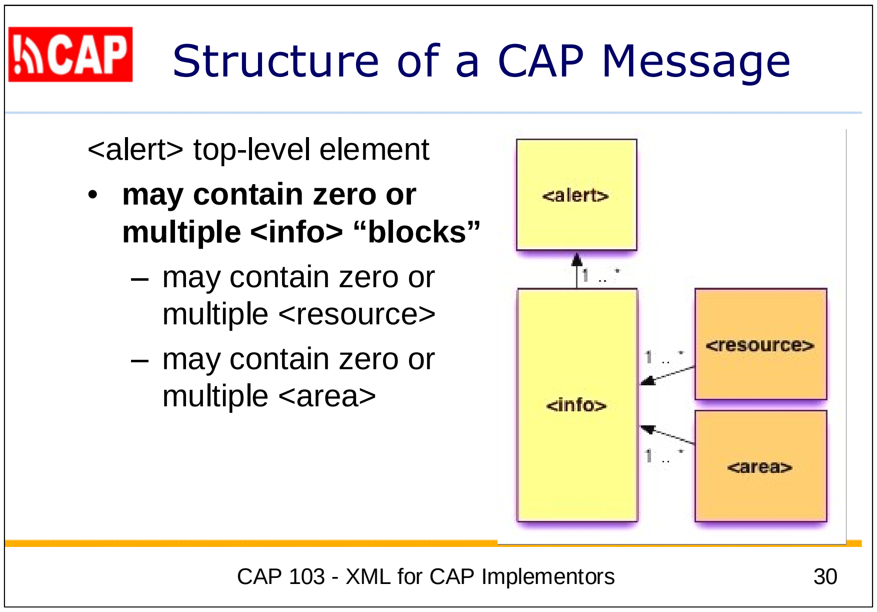
CAP 103 - XML for CAP Implementors 29

The next element is **code**, an optional text element that can occur multiple times within its parent *info* element.

Then we have the optional element named **note**. This element is intended for use with Cancel and Error messages alerting about the cancellation of an earlier message or an error in an earlier message.

The element named **references** is where we are able to identify the earlier message being referenced, such as those being cancelled or being alerted as erroneous. In the references element, such an earlier message is identified by stringing together with commas the values of *sender*, *identifier* and *sent* in the those earlier messages.
If there is more than one earlier message being referenced, those references must be separated from each other by a space.

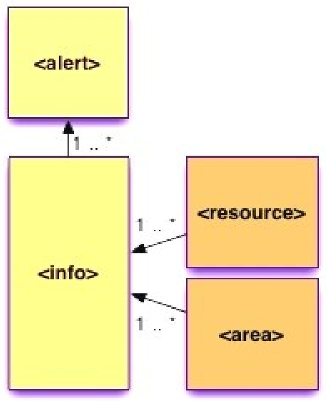
Now we see the element named **incidents**. This too is a grouping that can have multiple incident references, separated by a space.



Structure of a CAP Message

<alert> top-level element

- **may contain zero or multiple <info> “blocks”**
  - may contain zero or multiple <resource>
  - may contain zero or multiple <area>



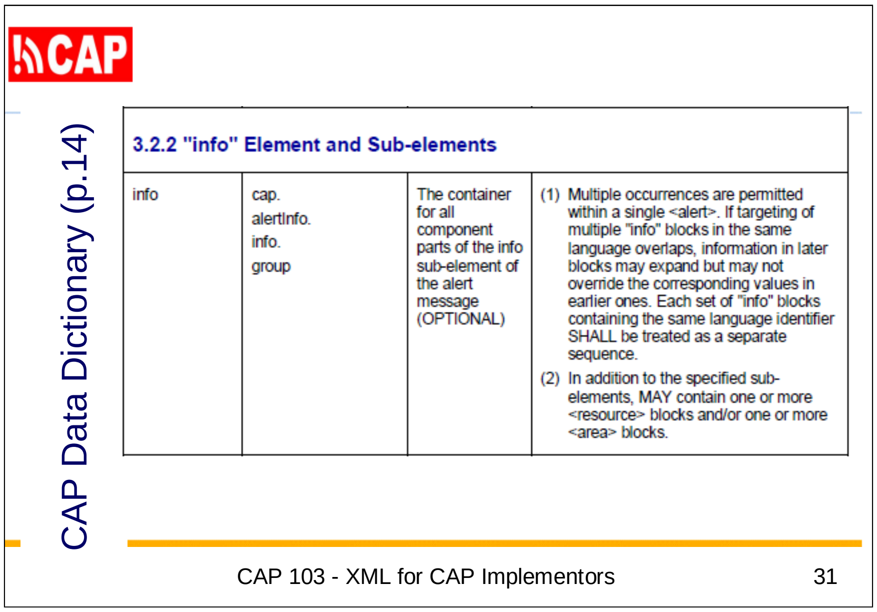
CAP 103 - XML for CAP Implementors 30

Let’s keep in mind the overall structure of a CAP alert message.

As represented in XML, the message always has an **alert** top-level element and we’ve just seen that there are several sub-elements available (some required and some optional).

We have seen that a CAP alert message may also contain **info** “blocks”. Let me point out that the term “block” is just an informal word for a grouping container which in XML is more properly simply another “element”.

Later, we’ll look at how any *info block* may contain zero or multiple <resource> elements and zero or multiple <area> elements as well.



CAP Data Dictionary (p.14)

3.2.2 "info" Element and Sub-elements

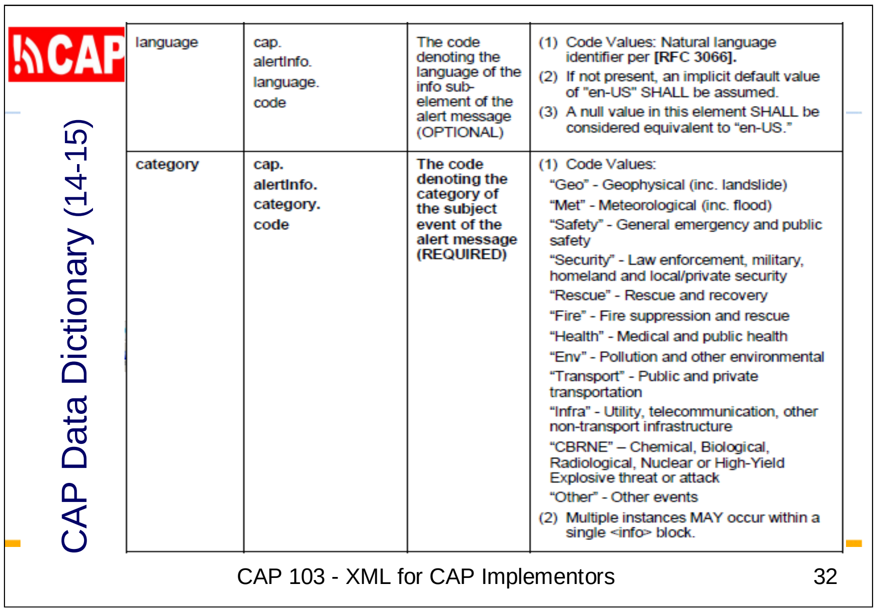
info	cap. alertInfo. info. group	The container for all component parts of the info sub-element of the alert message (OPTIONAL)	(1) Multiple occurrences are permitted within a single <alert>. If targeting of multiple "info" blocks in the same language overlaps, information in later blocks may expand but may not override the corresponding values in earlier ones. Each set of "info" blocks containing the same language identifier SHALL be treated as a separate sequence. (2) In addition to the specified sub-elements, MAY contain one or more <resource> blocks and/or one or more <area> blocks.

CAP 103 - XML for CAP Implementors 31

You can have make multiple info elements, as we have seen already.

The info element is optional, so it is valid in a CAP alert message to not have any info element.

There is a note here in the data dictionary about the handling of multiple languages across multiple info elements:
"If targeting of multiple "info" blocks in the same language overlaps, information in later blocks may expand but may not override the corresponding values in earlier ones. Each set of ‘info’ blocks containing the same language identifier SHALL be treated as a separate sequence."



CAP Data Dictionary (14-15)

language	cap. alertInfo. language. code	The code denoting the language of the info sub-element of the alert message (OPTIONAL)	(1) Code Values: Natural language identifier per **[RFC 3066].** (2) If not present, an implicit default value of "en-US" SHALL be assumed. (3) A null value in this element SHALL be considered equivalent to "en-US."
category	**cap. alertInfo. category. code**	**The code denoting the category of the subject event of the alert message (REQUIRED)**	(1) Code Values: "Geo" - Geophysical (inc. landslide) "Met" - Meteorological (inc. flood) "Safety" - General emergency and public safety "Security" - Law enforcement, military, homeland and local/private security "Rescue" - Rescue and recovery "Fire" - Fire suppression and rescue "Health" - Medical and public health "Env" - Pollution and other environmental "Transport" - Public and private transportation "Infra" - Utility, telecommunication, other non-transport infrastructure "CBRNE" – Chemical, Biological, Radiological, Nuclear or High-Yield Explosive threat or attack "Other" - Other events (2) Multiple instances MAY occur within a single <info> block.

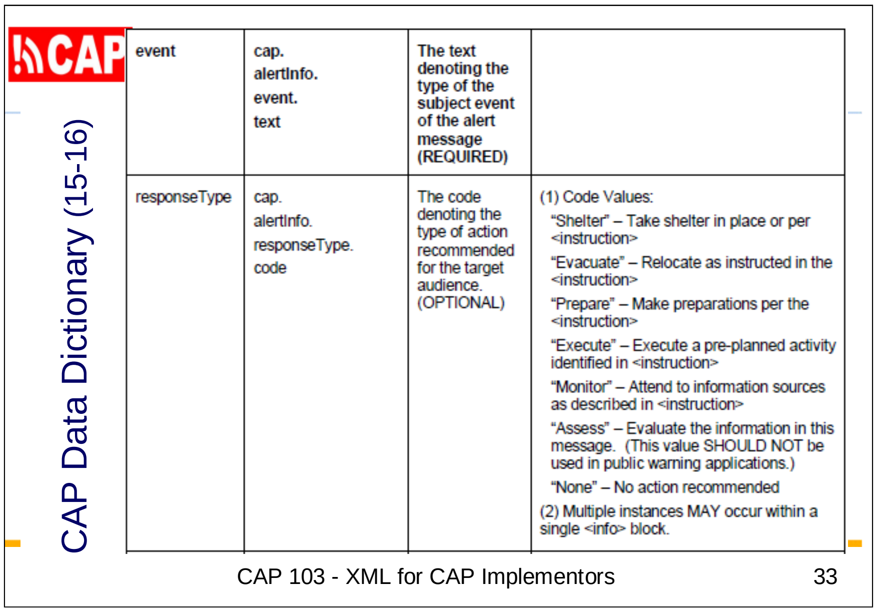
CAP 103 - XML for CAP Implementors 32

And now we see the **language** sub-element for the info element. The value in this element must be one of the common Internet language designators given in RFC 3066.

For instance, “en” indicates English while “fr” indicates French. Also, “en-US’ indicates English as used in the United States and “fr-CA” indicates French as used in Canada. There is an implicit default value of "en-US" in cases where this sub-element is missing or it has a null value (“is empty”).

The next element is named **category**. The element is required and there can be multiple instances. In each instance, the value of this element is restricted to be one of the following codes:

Geo, Met, Safety, Security, Rescue, Fire,
Health, Env, Transport, Infra, CBRNE, Other



CAP Data Dictionary (15-16)

event	**cap. alertInfo. event. text**	**The text denoting the type of the subject event of the alert message (REQUIRED)**	
responseType	cap. alertInfo. responseType. code	The code denoting the type of action recommended for the target audience. (OPTIONAL)	(1) Code Values: "Shelter" – Take shelter in place or per <instruction> "Evacuate" – Relocate as instructed in the <instruction> "Prepare" – Make preparations per the <instruction> "Execute" – Execute a pre-planned activity identified in <instruction> "Monitor" – Attend to information sources as described in <instruction> "Assess" – Evaluate the information in this message. (This value SHOULD NOT be used in public warning applications.) "None" – No action recommended (2) Multiple instances MAY occur within a single <info> block.

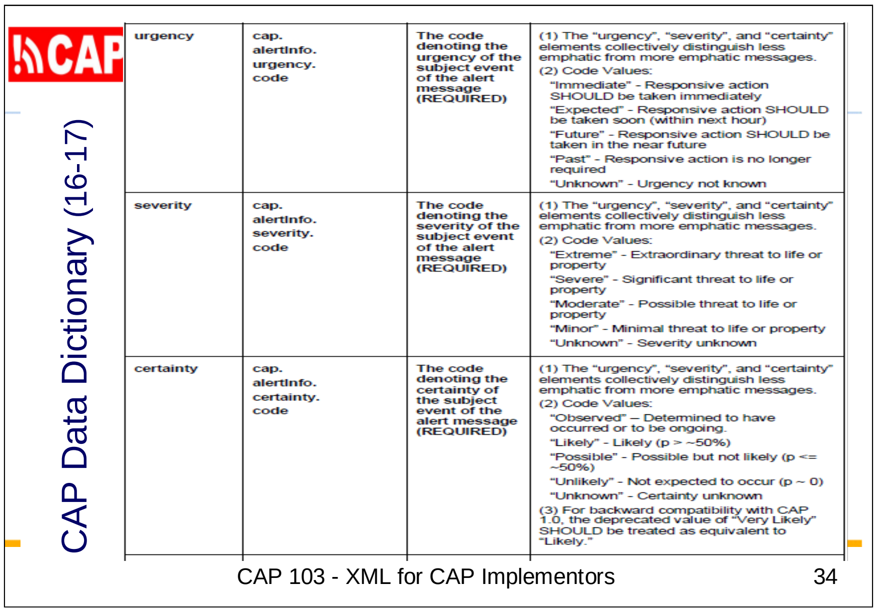
CAP 103 - XML for CAP Implementors 33

The **event** sub-element within the *info block* is required. Its text value is to be descriptive of the type of event being alerted.

The event value can be any text, but you may find that CAP Profiles provide suggested or required values.

Next we see responseType, an optional element with a fixed set of code values. The element can have multiple instances and each instance would be one of these codes:
Shelter, Evacuate, Prepare, Execute, Monitor, Assess, None

Note that the "Assess" value SHOULD NOT be used in public warning applications.



CAP Data Dictionary (16-17)

urgency	cap. alertInfo. urgency. code	**The code denoting the urgency of the subject event of the alert message (REQUIRED)**	(1) The "urgency", "severity", and "certainty" elements collectively distinguish less emphatic from more emphatic messages. (2) Code Values: "Immediate" - Responsive action SHOULD be taken immediately "Expected" - Responsive action SHOULD be taken soon (within next hour) "Future" - Responsive action SHOULD be taken in the near future "Past" - Responsive action is no longer required "Unknown" - Urgency not known
severity	cap. alertInfo. severity. code	**The code denoting the severity of the subject event of the alert message (REQUIRED)**	(1) The "urgency", "severity", and "certainty" elements collectively distinguish less emphatic from more emphatic messages. (2) Code Values: "Extreme" - Extraordinary threat to life or property "Severe" - Significant threat to life or property "Moderate" - Possible threat to life or property "Minor" - Minimal threat to life or property "Unknown" - Severity unknown
certainty	cap. alertInfo. certainty. code	**The code denoting the certainty of the subject event of the alert message (REQUIRED)**	(1) The "urgency", "severity", and "certainty" elements collectively distinguish less emphatic from more emphatic messages. (2) Code Values: "Observed" – Determined to have occurred or to be ongoing. "Likely" - Likely (p > ~50%) "Possible" - Possible but not likely (p <= ~50%) "Unlikely" - Not expected to occur (p ~ 0) "Unknown" - Certainty unknown (3) For backward compatibility with CAP 1.0, the deprecated value of "Very Likely" SHOULD be treated as equivalent to "Likely."

CAP 103 - XML for CAP Implementors 34

The next three elements are required: “urgency”, “severity”, and “certainty”. Collectively, these elements are to distinguish less emphatic from more emphatic messages.

There are five allowed code values in each of these three elements.

These are the code values for **urgency**:

Immediate, Expected, Future, Past, Unknown

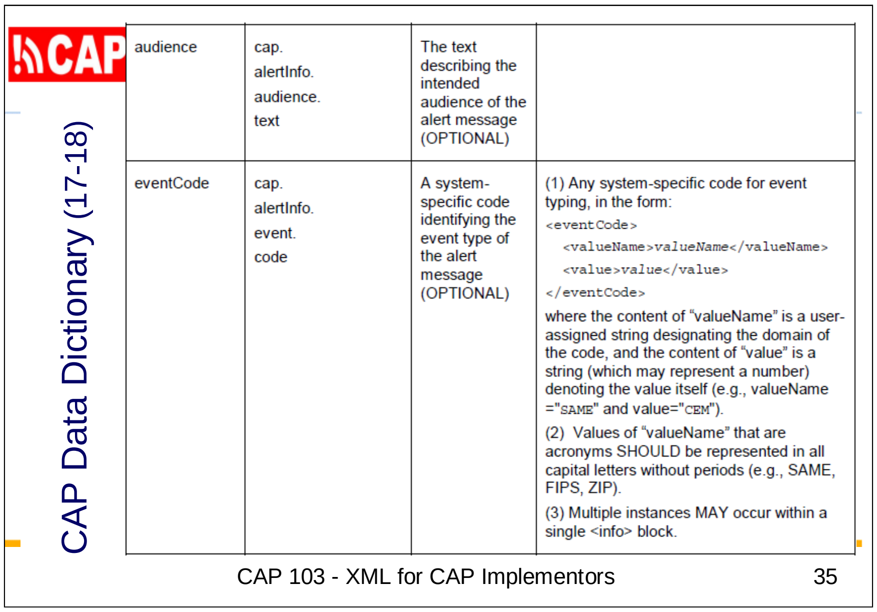
The code values for **severity** are:

Extreme, Severe, Moderate, Minor, Unknown

And, the code values for **certainty** are:

Observed, Likely, Possible, Unlikely, Unknown

It is also noted in CAP versions 1.1 and 1.2 that the deprecated value of “Very Likely” SHOULD be treated as equivalent to “Likely” for compatibility with version 1.0.



CAP Data Dictionary (17-18)

audience	cap. alertInfo. audience. text	The text describing the intended audience of the alert message (OPTIONAL)	

eventCode	cap.
	alertInfo.
	event.
	code
		A system-specific code identifying the event type of the alert message (OPTIONAL)	(1) Any system-specific code for event typing, in the form:
			`<eventCode>`
			`<valueName>valueName</valueName>`
			`<value>value</value>`
			`</eventCode>`
			where the content of "valueName" is a user-assigned string designating the domain of the code, and the content of "value" is a string (which may represent a number) denoting the value itself (e.g., valueName ="SAME" and value="CEM").
			(2) Values of "valueName" that are acronyms SHOULD be represented in all capital letters without periods (e.g., SAME, FIPS, ZIP).
			(3) Multiple instances MAY occur within a single <info> block.

CAP 103 - XML for CAP Implementors 35

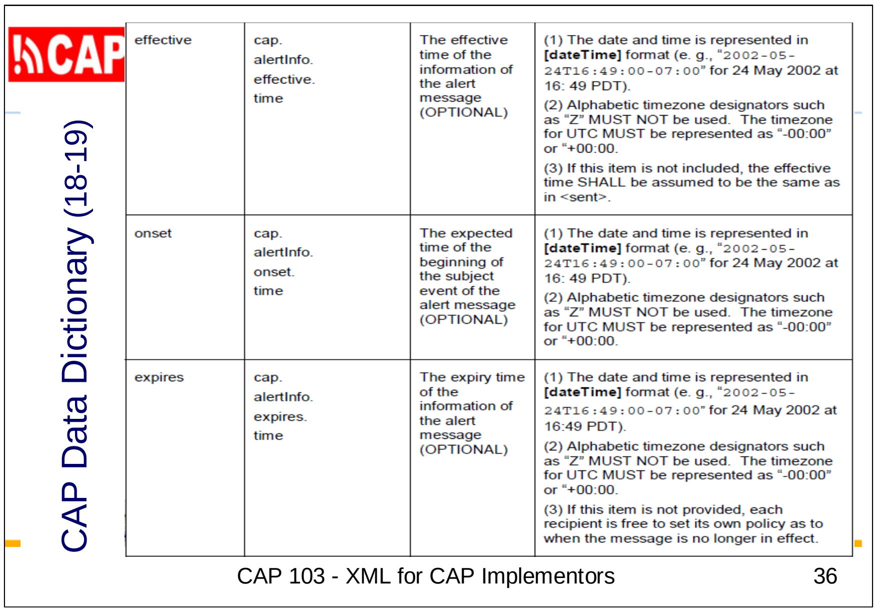
Here we see the optional text element **audience**, which would be descriptive of the intended audience of the alert message.

Next we have the optional element named **eventCode**. The value of each instance of this repeatable element is given in a "name-value" format.

To represent a named code and the value of that named code, it is necessary to use two sub-elements: **valueName** and **value**. Here is what that looks like:

```
<eventCode>
    <valueName>valueName</valueName>
    <value>value</value>
</eventCode>
```

It is also noted that the named code within valueName should be in All Capital letters if it is an "initialism" or acronym.



CAP Data Dictionary (18-19)

effective	cap. alertInfo. effective. time	The effective time of the information of the alert message (OPTIONAL)	(1) The date and time is represented in **[dateTime]** format (e. g., "2002-05-24T16:49:00-07:00" for 24 May 2002 at 16: 49 PDT). (2) Alphabetic timezone designators such as "Z" MUST NOT be used. The timezone for UTC MUST be represented as "-00:00" or "+00:00. (3) If this item is not included, the effective time SHALL be assumed to be the same as in <sent>.
onset	cap. alertInfo. onset. time	The expected time of the beginning of the subject event of the alert message (OPTIONAL)	(1) The date and time is represented in **[dateTime]** format (e. g., "2002-05-24T16:49:00-07:00" for 24 May 2002 at 16: 49 PDT). (2) Alphabetic timezone designators such as "Z" MUST NOT be used. The timezone for UTC MUST be represented as "-00:00" or "+00:00.
expires	cap. alertInfo. expires. time	The expiry time of the information of the alert message (OPTIONAL)	(1) The date and time is represented in **[dateTime]** format (e. g., "2002-05-24T16:49:00-07:00" for 24 May 2002 at 16:49 PDT). (2) Alphabetic timezone designators such as "Z" MUST NOT be used. The timezone for UTC MUST be represented as "-00:00" or "+00:00. (3) If this item is not provided, each recipient is free to set its own policy as to when the message is no longer in effect.

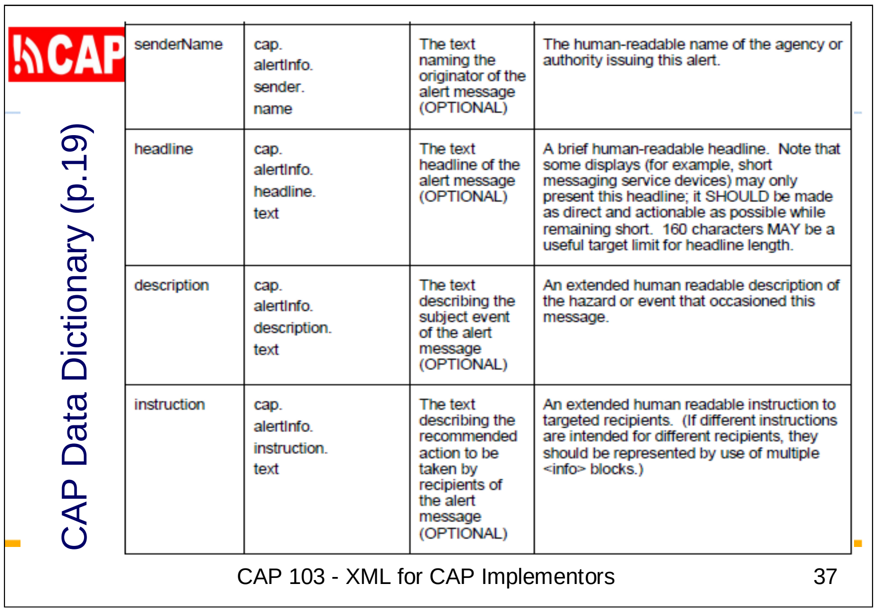
CAP 103 - XML for CAP Implementors 36

The next three elements, **effective, onset**, and **expires** are all optional and they are all in [dateTime] format, described earlier.

There is a note saying that if the *effective* value is not present, then the effective time of the alert shall be assumed to be the same as the value in the sent element (which is required to have a value).

There is no assumption made in the absence of an *onset* value.

If there is value for the expires element, the note says the recipient is free to "set its own policy as to when the message is no longer in effect".


CAP Data Dictionary (p.19)

senderName	cap. alertInfo. sender. name	The text naming the originator of the alert message (OPTIONAL)	The human-readable name of the agency or authority issuing this alert.
headline	cap. alertInfo. headline. text	The text headline of the alert message (OPTIONAL)	A brief human-readable headline. Note that some displays (for example, short messaging service devices) may only present this headline; it SHOULD be made as direct and actionable as possible while remaining short. 160 characters MAY be a useful target limit for headline length.
description	cap. alertInfo. description. text	The text describing the subject event of the alert message (OPTIONAL)	An extended human readable description of the hazard or event that occasioned this message.
instruction	cap. alertInfo. instruction. text	The text describing the recommended action to be taken by recipients of the alert message (OPTIONAL)	An extended human readable instruction to targeted recipients. (If different instructions are intended for different recipients, they should be represented by use of multiple <info> blocks.)

CAP 103 - XML for CAP Implementors 37

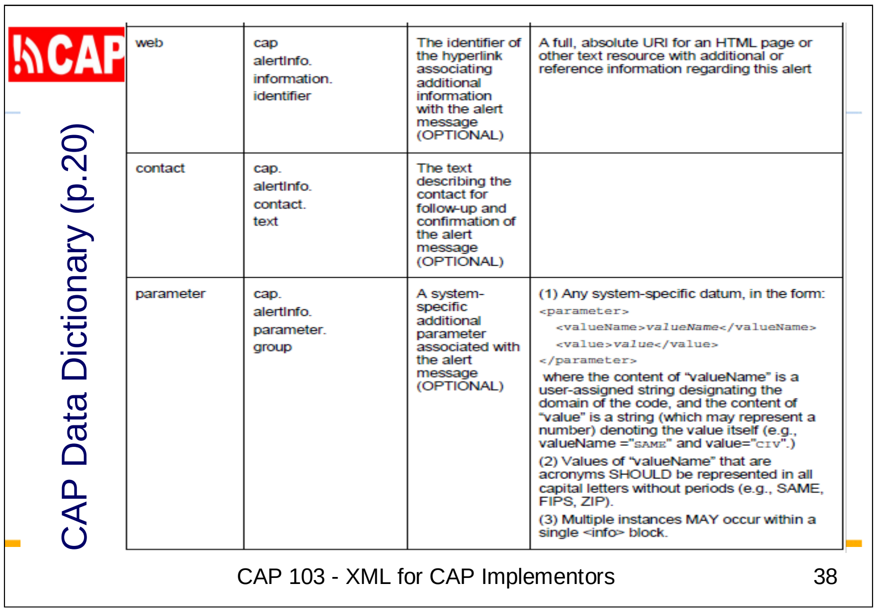
Here we have four optional elements, each intended to provide human-readable text.

The value of the element named **originator** is defined to contain the name of the agency or authority issuing this alert.

There is a note about the length of the value in the **headline** element. It says the headline SHOULD be made as direct and actionable as possible while remaining short. The length should be always less than 160 characters, in some cases less than 90 characters. It is also noted that some device displays may present only this value when alerting.

The value of the element named **description** is defined to contain a description of the hazard or event that occasioned this message.

The value of the element named **instruction** is defined to contain the recommended action to be taken by recipients of the alert message . It is also noted that multiple *info blocks* are to be used when different instructions are intended for multiple recipients.



CAP Data Dictionary (p.20)

web	cap alertInfo. information. identifier	The identifier of the hyperlink associating additional information with the alert message (OPTIONAL)	A full, absolute URI for an HTML page or other text resource with additional or reference information regarding this alert
contact	cap. alertInfo. contact. text	The text describing the contact for follow-up and confirmation of the alert message (OPTIONAL)	
parameter	cap. alertInfo. parameter. group	A system-specific additional parameter associated with the alert message (OPTIONAL)	(1) Any system-specific datum, in the form: `<parameter>` `<valueName>valueName</valueName>` `<value>value</value>` `</parameter>` where the content of "valueName" is a user-assigned string designating the domain of the code, and the content of "value" is a string (which may represent a number) denoting the value itself (e.g., valueName ="SAME" and value="CIV".) (2) Values of "valueName" that are acronyms SHOULD be represented in all capital letters without periods (e.g., SAME, FIPS, ZIP). (3) Multiple instances MAY occur within a single <info> block.

CAP 103 - XML for CAP Implementors 38

The next element is named **web**. This optional element points to additional or reference information regarding this alert. If present, the value must be formatted as an "absolute" URI (that is, use of a "relative" URI is not valid).

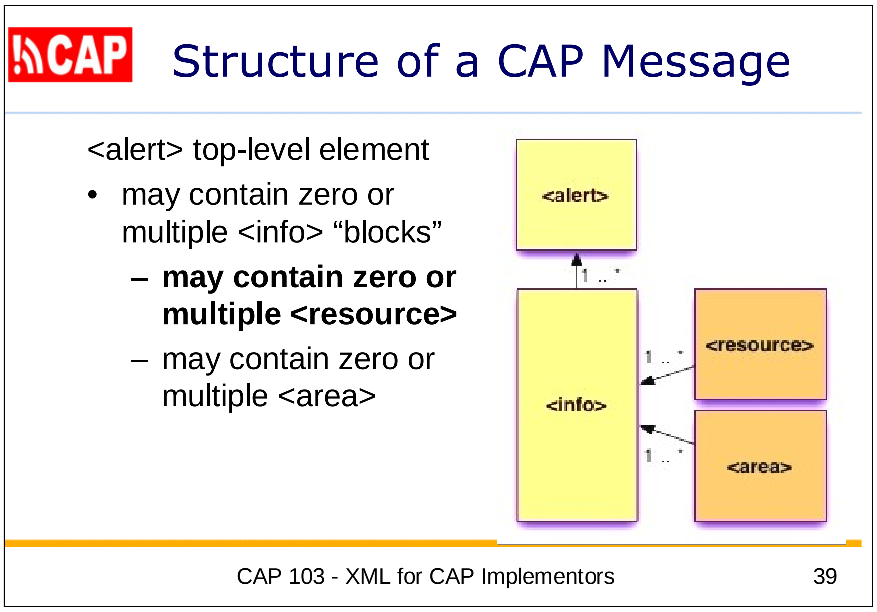
Then we have the optional element named **contact**. Its value has text describing the contact for follow-up and confirmation of the alert message.

The optional element named **parameter** provides in each instance a name-value pair encoding some kind of parameter associated with the alert message.

To represent a named code and the value of that named code, it is necessary to use two sub-elements: **valueName** and **value**. Here is what that looks like:

```
<parameter>
    <valueName>valueName</valueName>
    <value>value</value>
</parameter>
```

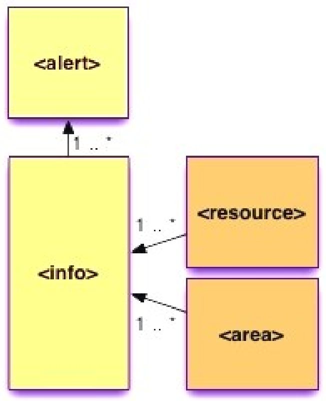
It is noted here also that the named code within valueName should be in All Capital letters if it is an "initialism" or acronym.



Structure of a CAP Message

<alert> top-level element

- may contain zero or multiple <info> "blocks"
  - **may contain zero or multiple <resource>**
  - may contain zero or multiple <area>

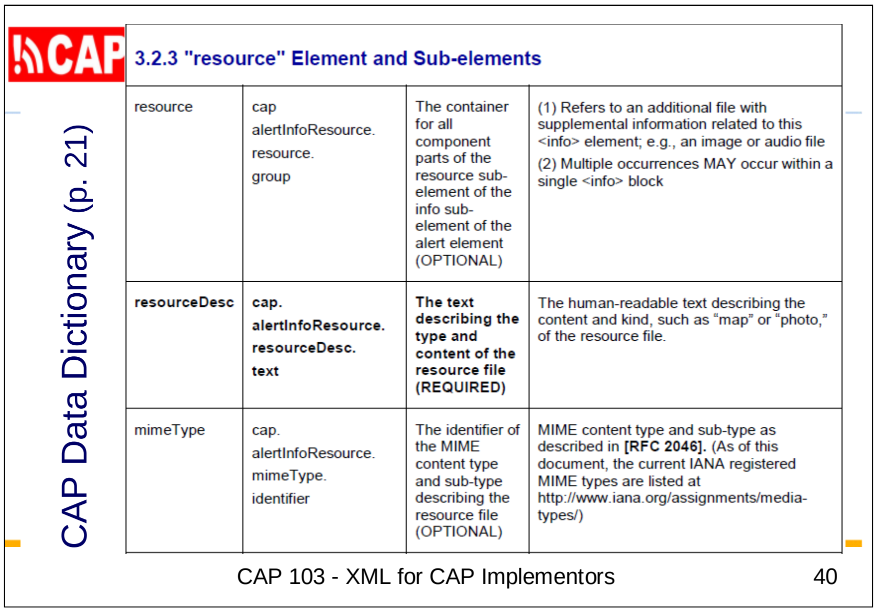


CAP 103 - XML for CAP Implementors 39

Again, here is the overall structure of a CAP alert message.

Now we are inside of the definition of an *info block*.

Let's look at the CAP data dictionary definition of the **resource** element.



CAP Data Dictionary (p. 21)

3.2.3 "resource" Element and Sub-elements

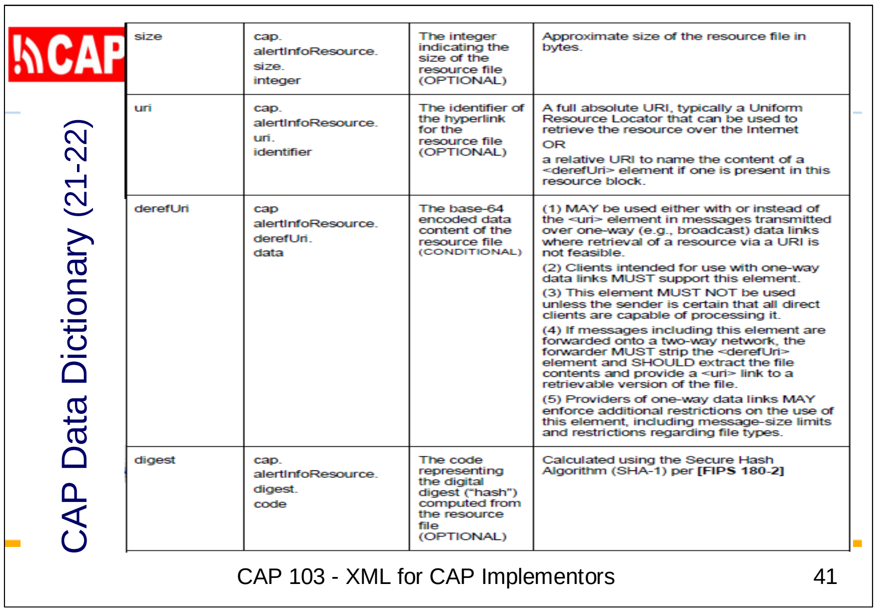
resource	cap alertInfoResource. resource. group	The container for all component parts of the resource sub-element of the info sub-element of the alert element (OPTIONAL)	(1) Refers to an additional file with supplemental information related to this <info> element; e.g., an image or audio file (2) Multiple occurrences MAY occur within a single <info> block
resourceDesc	**cap. alertInfoResource. resourceDesc. text**	**The text describing the type and content of the resource file (REQUIRED)**	The human-readable text describing the content and kind, such as "map" or "photo," of the resource file.
mimeType	cap. alertInfoResource. mimeType. identifier	The identifier of the MIME content type and sub-type describing the resource file (OPTIONAL)	MIME content type and sub-type as described in **[RFC 2046].** (As of this document, the current IANA registered MIME types are listed at http://www.iana.org/assignments/media-types/)

CAP 103 - XML for CAP Implementors 40

The optional grouping element named **resource** acts as the container for further sub-elements. An alert message may have multiple resource elements within the *info block*.

If there is a resource element, it will always contain at least the required element named **resourceDesc**. Its text value describes the type and content of the resource file, such as 'map' or 'photo'.

Also defined within the info/resource element is the optional element named **mimeType**. (MIME originated as Multipurpose Internet Mail Extensions). The value of the *resource* element identifies the MIME content type and sub-type describing the resource file. These MIME types are as defined [RFC 2046] and registered through IANA (Internet Assigned Number Authority).



CAP Data Dictionary (21-22)

size	cap. alertInfoResource. size. integer	The integer indicating the size of the resource file (OPTIONAL)	Approximate size of the resource file in bytes.
uri	cap. alertInfoResource. uri. identifier	The identifier of the hyperlink for the resource file (OPTIONAL)	A full absolute URI, typically a Uniform Resource Locator that can be used to retrieve the resource over the Internet OR a relative URI to name the content of a <derefUri> element if one is present in this resource block.
derefUri	cap alertInfoResource. derefUri. data	The base-64 encoded data content of the resource file (CONDITIONAL)	(1) MAY be used either with or instead of the <uri> element in messages transmitted over one-way (e.g., broadcast) data links where retrieval of a resource via a URI is not feasible. (2) Clients intended for use with one-way data links MUST support this element. (3) This element MUST NOT be used unless the sender is certain that all direct clients are capable of processing it. (4) If messages including this element are forwarded onto a two-way network, the forwarder MUST strip the <derefUri> element and SHOULD extract the file contents and provide a <uri> link to a retrievable version of the file. (5) Providers of one-way data links MAY enforce additional restrictions on the use of this element, including message-size limits and restrictions regarding file types.
digest	cap. alertInfoResource. digest. code	The code representing the digital digest ("hash") computed from the resource file (OPTIONAL)	Calculated using the Secure Hash Algorithm (SHA-1) per **[FIPS 180-2]**

CAP 103 - XML for CAP Implementors 41

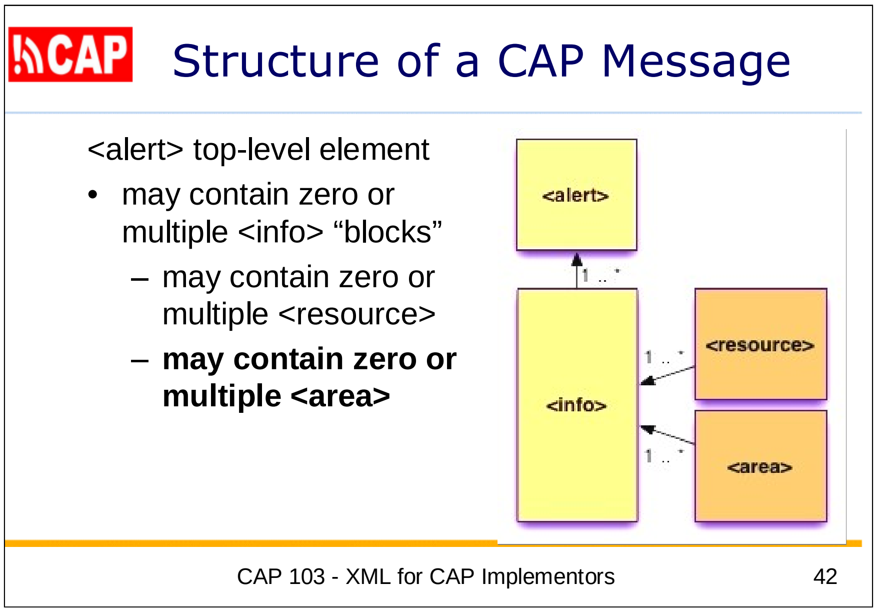
Also within the info/resource element we find the optional element named **size**. Its value is an integer indicating the approximate size of the resource file in bytes.

Next within info/resource is the optional element named **uri**. Its value identifies the hyperlink for the resource file, given as an URI. Typically, the value is an absolute URL (Uniform Resource Locator) that can be used to retrieve the resource over the Internet. Its value can also be a relative URI naming the content of a *derefUri* element if one is present in this resource block.

Next is the conditional element named **derefUri**, which is short for "de-referenced URI". The idea here is that it is useful for an alert to include some information that would usually be accessed via a URI pointing to a resource such as a photo, video, map, etc. If the alert recipient can access that referenced resource, then it is sufficient to provide the URI that points to it. However, when alert recipients cannot use a URI to obtain the referenced information, then the content must be embedded in the alert itself. This is commonly the case with one-way (e.g., broadcast) data links.

There are further notes in the CAP data dictionary concerning use of this element, but we'll not cover those in the presentation.

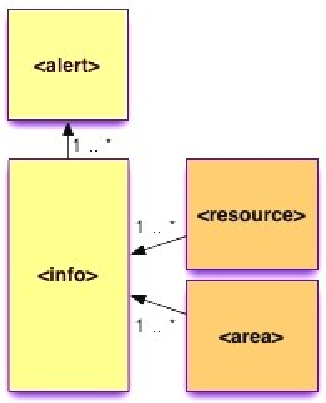
I will move now to the optional element named **digest**. Its value represents the digital digest ('hash') computed from the resource file using the Secure Hash Algorithm (SHA-1) per FIPS 180-2.



Structure of a CAP Message

<alert> top-level element

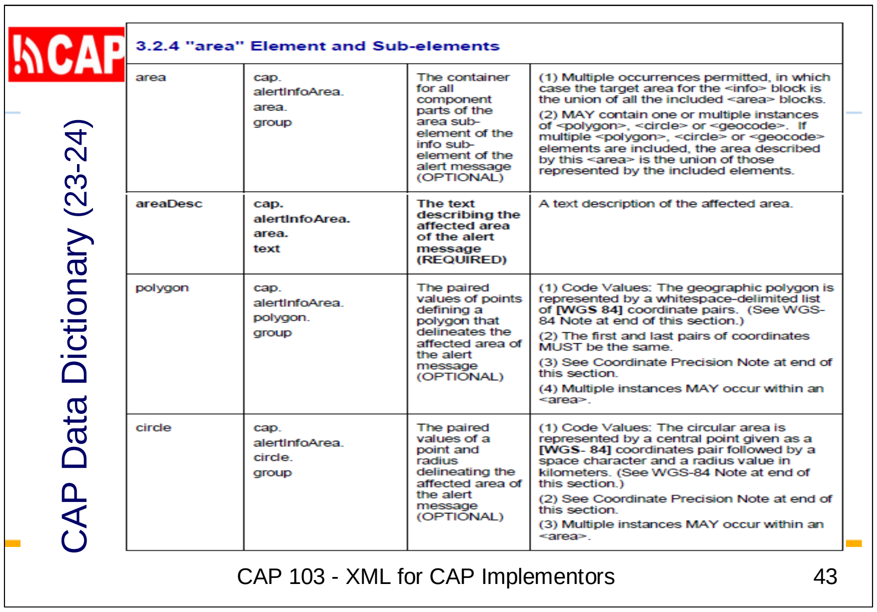
- may contain zero or multiple <info> "blocks"
  - may contain zero or multiple <resource>
  - **may contain zero or multiple <area>**



CAP 103 - XML for CAP Implementors 42

We are still inside the definition of an *info block*.

Now we can finish up by looking at how the CAP data dictionary defines the **area** element.



CAP Data Dictionary (23-24)

3.2.4 "area" Element and Sub-elements

area	cap. alertInfoArea. area. group	The container for all component parts of the area sub-element of the info sub-element of the alert message (OPTIONAL)	(1) Multiple occurrences permitted, in which case the target area for the <info> block is the union of all the included <area> blocks. (2) MAY contain one or multiple instances of <polygon>, <circle> or <geocode>. If multiple <polygon>, <circle> or <geocode> elements are included, the area described by this <area> is the union of those represented by the included elements.
areaDesc	**cap. alertInfoArea. area. text**	**The text describing the affected area of the alert message (REQUIRED)**	A text description of the affected area.
polygon	cap. alertInfoArea. polygon. group	The paired values of points defining a polygon that delineates the affected area of the alert message (OPTIONAL)	(1) Code Values: The geographic polygon is represented by a whitespace-delimited list of **[WGS 84]** coordinate pairs. (See WGS-84 Note at end of this section.) (2) The first and last pairs of coordinates MUST be the same. (3) See Coordinate Precision Note at end of this section. (4) Multiple instances MAY occur within an <area>.
circle	cap. alertInfoArea. circle. group	The paired values of a point and radius delineating the affected area of the alert message (OPTIONAL)	(1) Code Values: The circular area is represented by a central point given as a **[WGS- 84]** coordinates pair followed by a space character and a radius value in kilometers. (See WGS-84 Note at end of this section.) (2) See Coordinate Precision Note at end of this section. (3) Multiple instances MAY occur within an <area>.

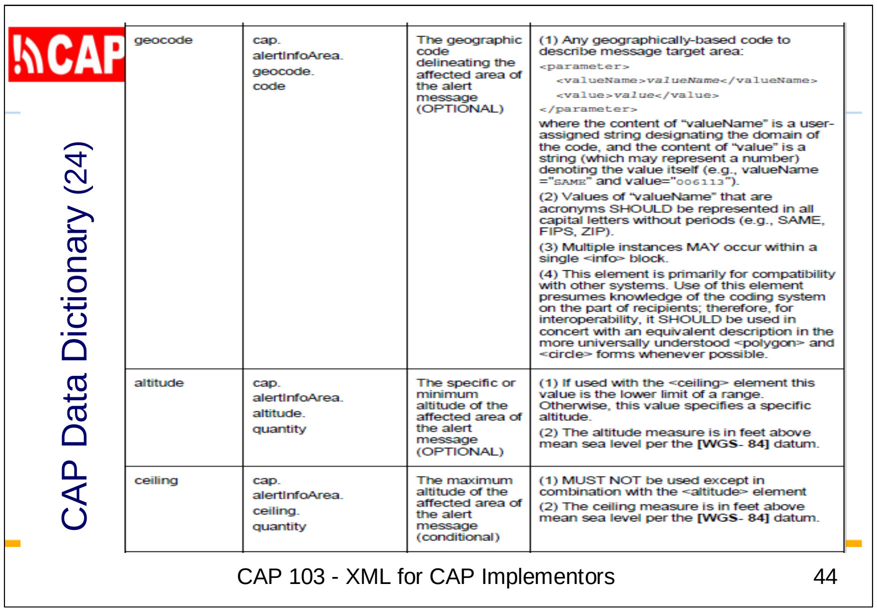
CAP 103 - XML for CAP Implementors 43

Next is the optional grouping element named **area**. It acts as the container for further sub-elements and an alert message may have multiple area elements within the *info block*. If there are multiple area elements, it is noted that the target area for the info blocks is the union of all the included <area> blocks, which themselves may be multiple instances of *polygon*, *circle* or *geocode*.

Within the *info/area* element, we always find an element named **areaDesc**. This element is required if the alert message contains an area element. Its value, in human-readable text, describes the affected area of the alert message.

The area can be further described by the optional element named **polygon**, and there can be multiple polygons within an info/area block. Its value contains coordinates at the points of a polygon surrounding the affected area of the alert message. Each point is given as a coordinate pair, in the form of latitude (comma) longitude. Each coordinate is stated in decimal degrees, signed negative in the Western and Southern hemispheres. A space separates each point (each pair of coordinates). Also, the first and last pairs of coordinates MUST be the same, which is to say the polygon must be closed.

There is also the optional element named **circle**, delineating the affected area of the alert message. There can be multiple circles within an info/area block, and each instance of a circle is represented as the paired value of a center point and a radius. The center point is a coordinate, in latitude (comma) longitude form as just described. A space follows the center point and that is followed by the radius value, in kilometers.



CAP Data Dictionary (24)

geocode	cap. alertInfoArea. geocode. code	The geographic code delineating the affected area of the alert message (OPTIONAL)	(1) Any geographically-based code to describe message target area: `<parameter>` `<valueName>valueName</valueName>` `<value>value</value>` `</parameter>` where the content of “valueName” is a user-assigned string designating the domain of the code, and the content of “value” is a string (which may represent a number) denoting the value itself (e.g., valueName ="SAME" and value="006113"). (2) Values of “valueName” that are acronyms SHOULD be represented in all capital letters without periods (e.g., SAME, FIPS, ZIP). (3) Multiple instances MAY occur within a single <info> block. (4) This element is primarily for compatibility with other systems. Use of this element presumes knowledge of the coding system on the part of recipients; therefore, for interoperability, it SHOULD be used in concert with an equivalent description in the more universally understood <polygon> and <circle> forms whenever possible.
altitude	cap. alertInfoArea. altitude. quantity	The specific or minimum altitude of the affected area of the alert message (OPTIONAL)	(1) If used with the <ceiling> element this value is the lower limit of a range. Otherwise, this value specifies a specific altitude. (2) The altitude measure is in feet above mean sea level per the **[WGS- 84]** datum.
ceiling	cap. alertInfoArea. ceiling. quantity	The maximum altitude of the affected area of the alert message (conditional)	(1) MUST NOT be used except in combination with the <altitude> element (2) The ceiling measure is in feet above mean sea level per the **[WGS- 84]** datum.

CAP 103 - XML for CAP Implementors 44

Also within the *info/area* element is the optional element named **geocode**, which can occur multiple times. Each instance gives a geographically-based code describing the message target area.
As we have seen before, it is necessary to use two sub-elements (valueName and value) to represent a named code and the value of that named code. Here is what it looks like:

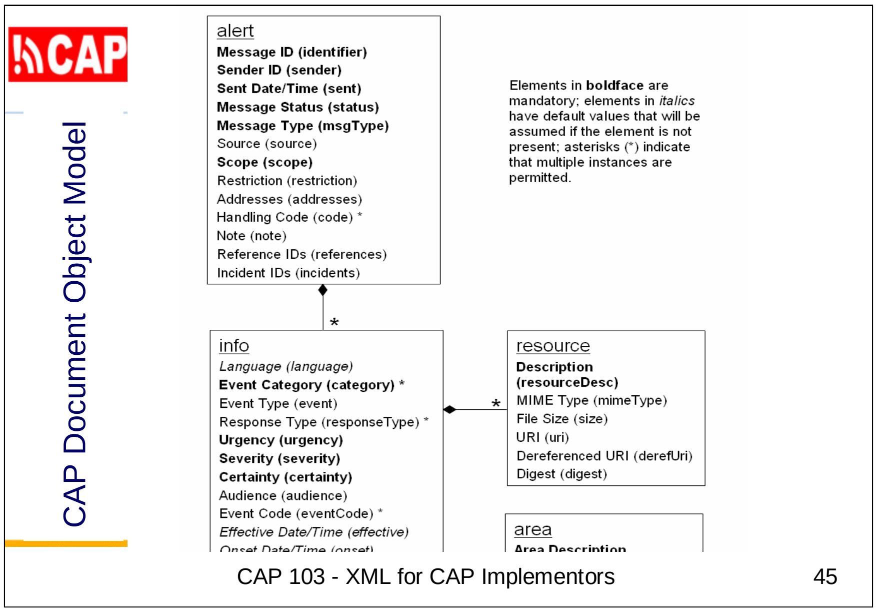
```
<geocode>
    <valueName>valueName</valueName>
    <value>value</value>
</geocode>
```

It is noted here also that the named code within valueName should be in All Capital letters without periods if they are acronyms (initialisms).

It is noted use of the *geocode* element presumes that recipients have knowledge of the coding system. The note states that *geocode* SHOULD be used in concert with an equivalent description in the more universally understood <polygon> and <circle> forms.

The next element is also optional and is named **altitude**. Its value gives the altitude in feet above mean sea level for the affected space of the alert message. If *altitude* is used with the *ceiling* element, the value in *altitude* is the lower limit of a range. Otherwise, the value specifies a specific altitude.

The last element defined in the CAP data dictionary is named **ceiling** and it is only used in combination with the altitude element. Its value gives the maximum altitude of the vertical range of the affected space of the alert message. It too is expressed in feet above mean sea level.



CAP Document Object Model

alert
Message ID (identifier)
Sender ID (sender)
Sent Date/Time (sent)
Message Status (status)
Message Type (msgType)
Source (source)
Scope (scope)
Restriction (restriction)
Addresses (addresses)
Handling Code (code) *
Note (note)
Reference IDs (references)
Incident IDs (incidents)

Elements in **boldface** are mandatory; elements in *italics* have default values that will be assumed if the element is not present; asterisks (*) indicate that multiple instances are permitted.

info
Language (language)
Event Category (category) *
Event Type (event)
Response Type (responseType) *
Urgency (urgency)
Severity (severity)
Certainty (certainty)
Audience (audience)
Event Code (eventCode) *
Effective Date/Time (effective)
Onset Date/Time (onset)

resource
Description (resourceDesc)
MIME Type (mimeType)
File Size (size)
URI (uri)
Dereferenced URI (derefUri)
Digest (digest)

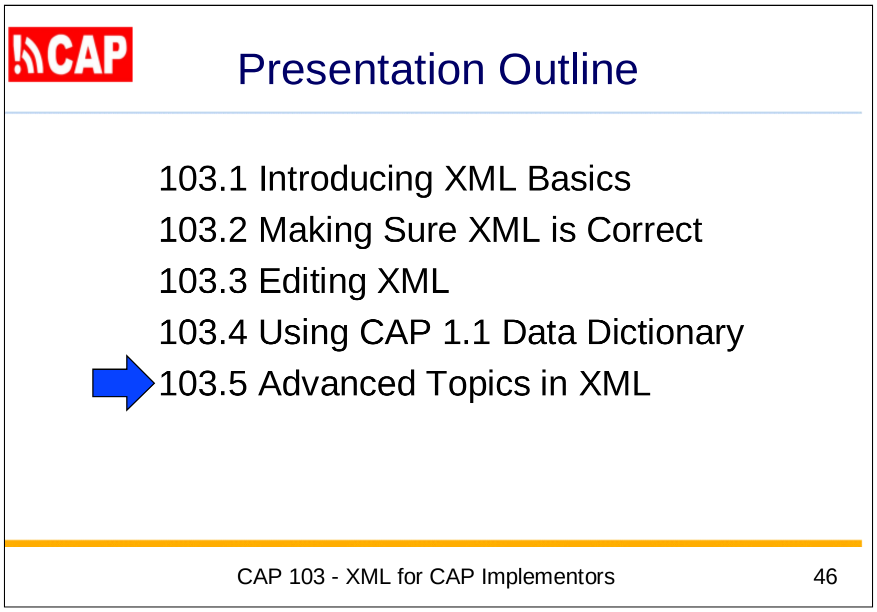
area
Area Description

CAP 103 - XML for CAP Implementors 45

Now we have looked at all of the elements in CAP version 1.1.

The full set of elements may seem complicated, but a CAP message in practice can be simple and most alerts are likely straightforward.

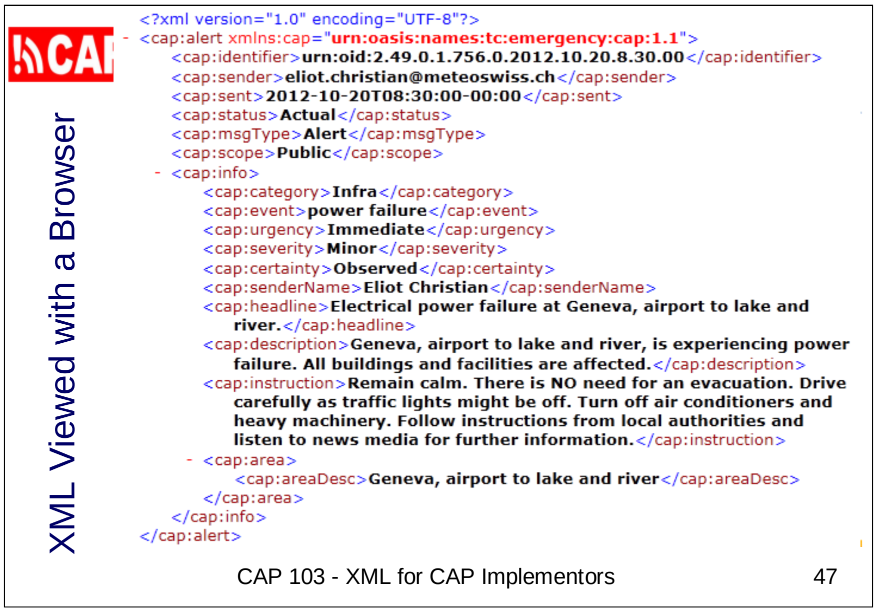
In the extreme case, a valid CAP alert can have as few as just the six required elements (with no info block).



Presentation Outline

CAP 103 - XML for CAP Implementors 46

The last topic of this session will introduce some Advanced Topics in XML



XML Viewed with a Browser

```
<?xml version="1.0" encoding="UTF-8"?>
- <cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.1">
    <cap:identifier>urn:oid:2.49.0.1.756.0.2012.10.20.8.30.00</cap:identifier>
    <cap:sender>eliot.christian@meteoswiss.ch</cap:sender>
    <cap:sent>2012-10-20T08:30:00-00:00</cap:sent>
    <cap:status>Actual</cap:status>
    <cap:msgType>Alert</cap:msgType>
    <cap:scope>Public</cap:scope>
  - <cap:info>
      <cap:category>Infra</cap:category>
      <cap:event>power failure</cap:event>
      <cap:urgency>Immediate</cap:urgency>
      <cap:severity>Minor</cap:severity>
      <cap:certainty>Observed</cap:certainty>
      <cap:senderName>Eliot Christian</cap:senderName>
      <cap:headline>Electrical power failure at Geneva, airport to lake and
        river.</cap:headline>
      <cap:description>Geneva, airport to lake and river, is experiencing power
        failure. All buildings and facilities are affected.</cap:description>
      <cap:instruction>Remain calm. There is NO need for an evacuation. Drive
        carefully as traffic lights might be off. Turn off air conditioners and
        heavy machinery. Follow instructions from local authorities and
        listen to news media for further information.</cap:instruction>
    - <cap:area>
        <cap:areaDesc>Geneva, airport to lake and river</cap:areaDesc>
      </cap:area>
    </cap:info>
  </cap:alert>
```

CAP 103 - XML for CAP Implementors 47

Here is the CAP alert as visitors might see it, in its “raw XML” form.

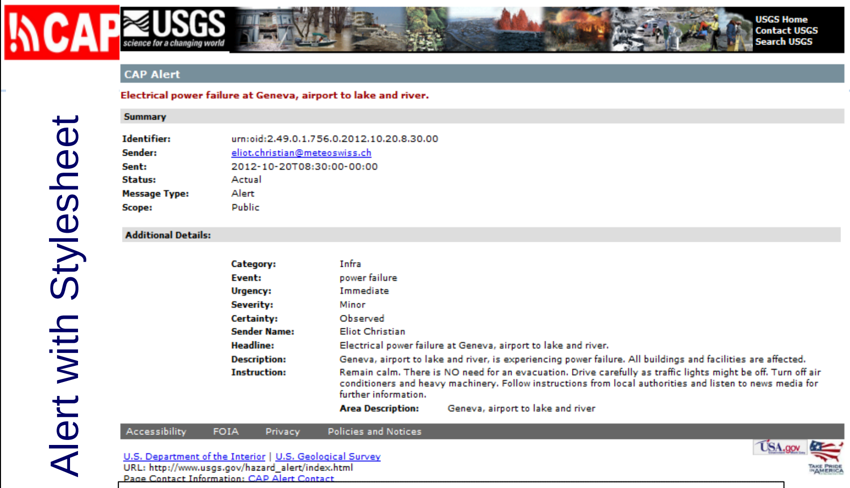
If the XML were not well-formed, the visitor would get an error. Otherwise, a visitor using Microsoft Internet Explorer sees a well-formed XML document displayed with color-coding that differentiates the XML tags from the element values.

The structure is handled using a plus or minus sign to the left of the elements. These can be clicked to expand or collapse the element structure.

A visitor using Safari would see only the element values. To view the raw XML, the visitor would have to right click and select "View Source“.

Clearly, the raw XML is not what you want your Web site visitors to see.

Alert with Stylesheet



CAP Alert

Electrical power failure at Geneva, airport to lake and river.

Summary

Identifier:	urn:oid:2.49.0.1.756.0.2012.10.20.8.30.00
Sender:	eliot.christian@meteoswiss.ch
Sent:	2012-10-20T08:30:00-00:00
Status:	Actual
Message Type:	Alert
Scope:	Public

Additional Details:

Category:	Infra
Event:	power failure
Urgency:	Immediate
Severity:	Minor
Certainty:	Observed
Sender Name:	Eliot Christian
Headline:	Electrical power failure at Geneva, airport to lake and river.
Description:	Geneva, airport to lake and river, is experiencing power failure. All buildings and facilities are affected.
Instruction:	Remain calm. There is NO need for an evacuation. Drive carefully as traffic lights might be off. Turn off air conditioners and heavy machinery. Follow instructions from local authorities and listen to news media for further information.
Area Description:	Geneva, airport to lake and river

Accessibility FOIA Privacy Policies and Notices

U.S. Department of the Interior | U.S. Geological Survey
URL: http://www.usgs.gov/hazard_alert/index.html
Page Contact Information: CAP Alert Contact

http://www.usgs.gov/hazard_alert/alerts/cap_alert.xsl

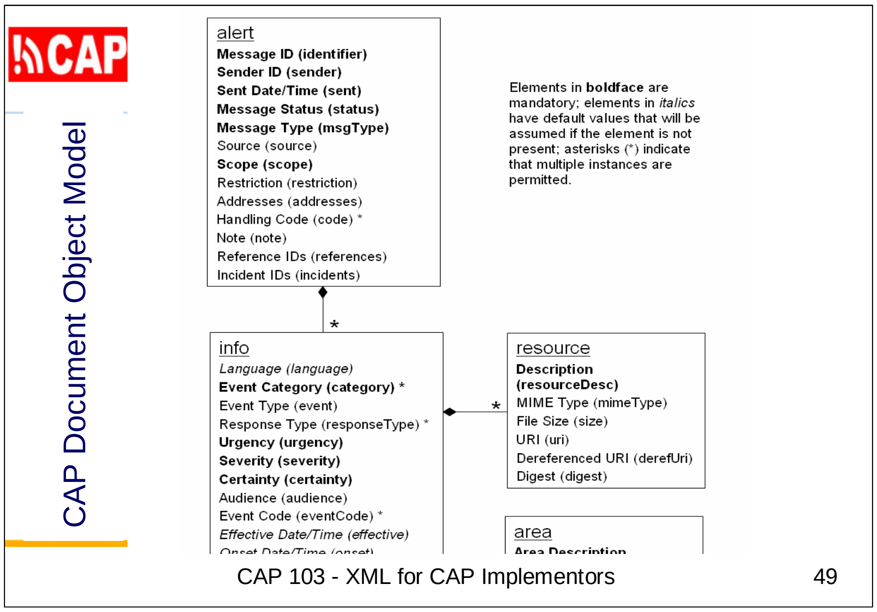
CAP 103 - XML for CAP Implementors 48

To make a CAP alert easier for a human visitor to read, we add a line to the CAP alert to reference an XML Stylesheet, called “XSLT”.

XSLT allows us to transform an XML document into HTML. Yet, the XML representation is still there so it can used by other processes, such as alert aggregators, text-to-speech processors, and so on.

Here I’ve applied a USGS stylesheet that I wrote years ago. You can see this makes the CAP alert nicely formatted for users who see the alert through a Web browser.

We don’t have time to cover the creation of XML spreadsheets in this course, but you may find it’s not too hard to use this example and customize it for your own site.

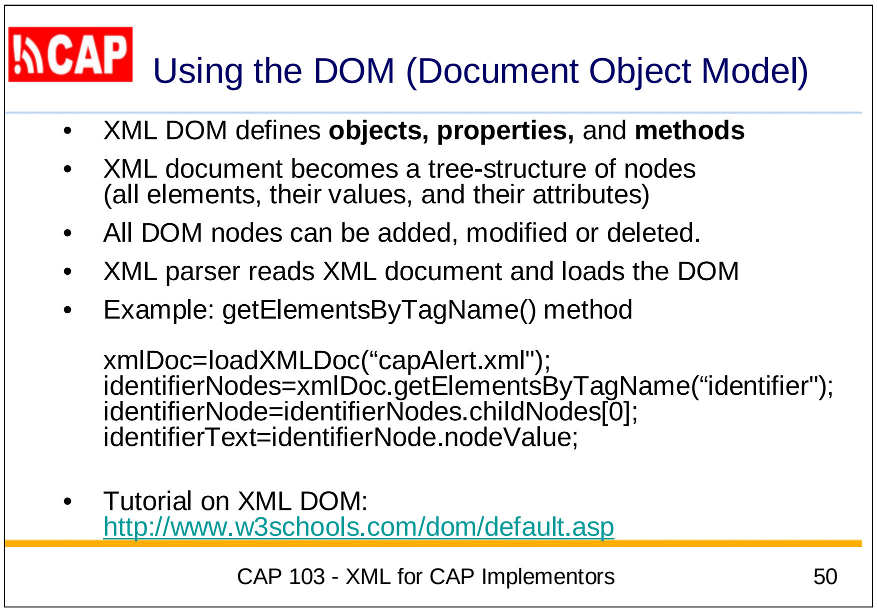


Elements in **boldface** are mandatory; elements in *italics* have default values that will be assumed if the element is not present; asterisks (*) indicate that multiple instances are permitted.

Ideally, a stylesheet should anticipate all of the CAP elements defined in the XML Schema for the CAP version you are using.

Here again is a simplified view of the CAP Document Object Model, showing just the overall structure and the names of defined elements.

Now we’ll look at how the Document Object Model is used at the programming level.



Using the DOM (Document Object Model)

- XML DOM defines **objects, properties,** and **methods**
- XML document becomes a tree-structure of nodes (all elements, their values, and their attributes)
- All DOM nodes can be added, modified or deleted.
- XML parser reads XML document and loads the DOM
- Example: getElementsByTagName() method

```
xmlDoc=loadXMLDoc(“capAlert.xml");
identifierNodes=xmlDoc.getElementsByTagName(“identifier");
identifierNode=identifierNodes.childNodes[0];
identifierText=identifierNode.nodeValue;
```

- Tutorial on XML DOM: http://www.w3schools.com/dom/default.asp

CAP 103 - XML for CAP Implementors 50

The XML DOM (Document Object Model) defines a standard way for accessing and manipulating an XML document. It defines the **objects and properties** of all XML elements, and the **methods** (interface) to access them.

The DOM treats the XML document as a tree-structure with all elements, their values, and their attributes referenced as “nodes”.
All DOM nodes can be accessed and new elements can be inserted.
Values of text and attributes of elements can be modified or deleted.

First, an XML parser needs to read an XML document and load it into an XML DOM object. Most browsers have a built-in XML parser and it produces an XML DOM that can be accessed with JavaScript.

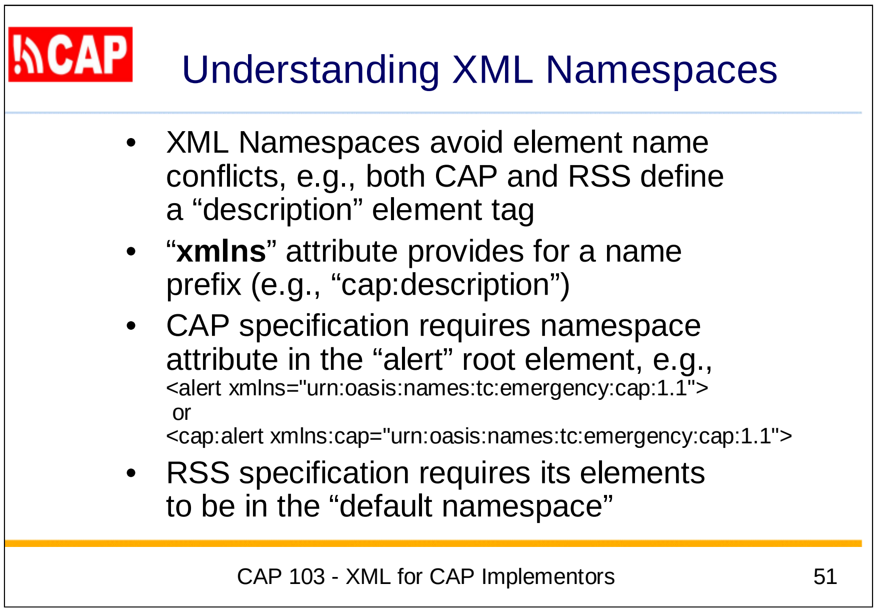
For example, the *getElementsByTagName()* method returns a node list containing all elements with the specified tag name, in the same order as they appear in the source document.

Here we load the DOM: **xmlDoc=loadXMLDoc(“capAlert.xml")**;
Then we get all of the nodes tagged as “identifier”
identifierNodes=xmlDoc.getElementsByTagName(“identifier");

We know identifier is required and there must be only one, so:
identifierNode=identifierNodes.childNodes[0];
Now we have the entire identifier element, including nodes for its tag, its attributes, its children, its siblings, its parent, and so on.
The node we want to get now is its text value, so we code:
identifierText=identifierNode.nodeValue;

If you have done any programming, navigating a DOM is fairly intuitive, although there are certainly some tricks to be learned.

We don’t have time in this session to further explore the XML DOM. There is a tutorial on this topic at **www.w3schools.com/dom**



Understanding XML Namespaces

- XML Namespaces avoid element name conflicts, e.g., both CAP and RSS define a “description” element tag
- “**xmlns**” attribute provides for a name prefix (e.g., “cap:description”)
- CAP specification requires namespace attribute in the “alert” root element, e.g.,
  <alert xmlns="urn:oasis:names:tc:emergency:cap:1.1">
  or
  <cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.1">
- RSS specification requires its elements to be in the “default namespace”

CAP 103 - XML for CAP Implementors 51

XML Namespaces provide a method to avoid element name conflicts—a situation where a tag name in an XML document would have multiple definitions because that tag was used in different schemas that the document draws content from.

For instance, we see that CAP has a “description” element tag and RSS also has a “description” element tag.

To tell them apart, we can put a separate name prefix on the cap tags, following the XML Namespaces rules. The tag would become “cap:description”.

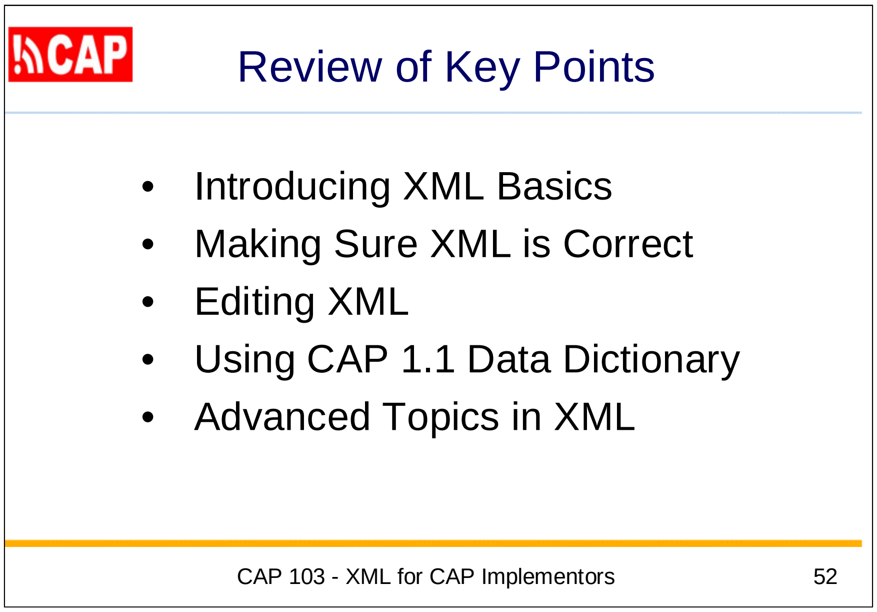
Of course, the XML parser needs to be told about this name prefix. We accomplish that with the “xmlns” attribute.

The CAP specification has a specific requirement about the namespace attribute. For the “alert” element, the specification says that it “MUST include the xmlns attribute referencing the CAP URN as the namespace, e.g.:
<alert xmlns="urn:oasis:names:tc:emergency:cap:1.1"> or

<cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.1">

In the case of the RSS specification, the requirement is that RSS elements use the “default namespace”. That is, the RSS tags do not have a namespace prefix.

In an RSS document, you can include any other elements you like as long as you provide an appropriate namespace attribute and use the corresponding prefixes.



Review of Key Points

- Introducing XML Basics
- Making Sure XML is Correct
- Editing XML
- Using CAP 1.1 Data Dictionary
- Advanced Topics in XML

CAP 103 - XML for CAP Implementors 52

Review of Key Points

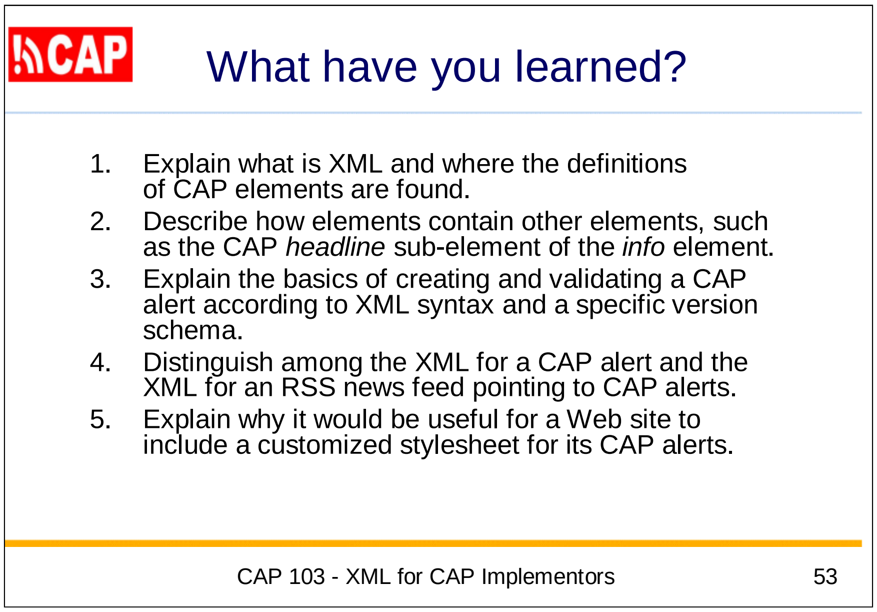
• Introducing XML Basics

• Making Sure XML is Correct

• Editing XML

• Advanced Topics in XML

• Sharing News Feeds across Organizations



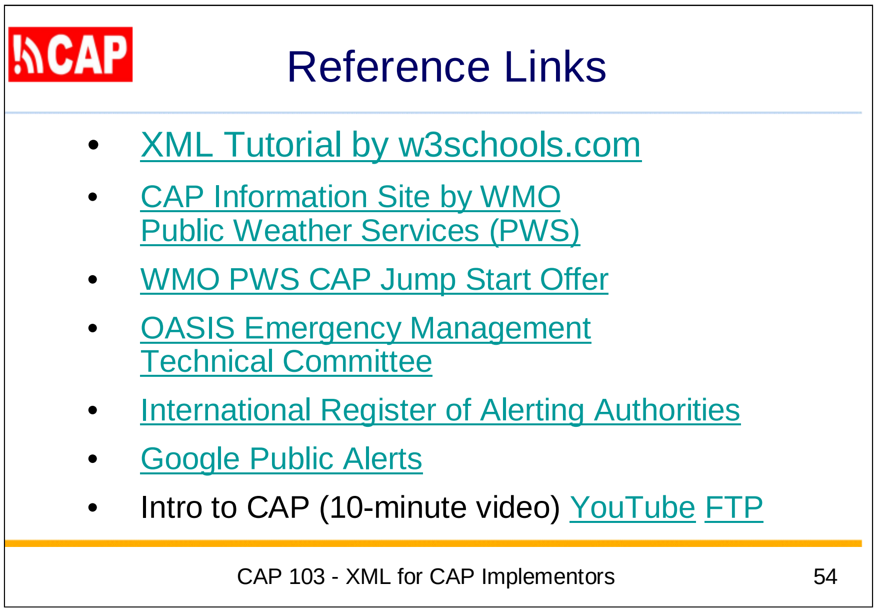
What have you learned?

1. Explain what is XML and where the definitions of CAP elements are found.
2. Describe how elements contain other elements, such as the CAP *headline* sub-element of the *info* element.
3. Explain the basics of creating and validating a CAP alert according to XML syntax and a specific version schema.
4. Distinguish among the XML for a CAP alert and the XML for an RSS news feed pointing to CAP alerts.
5. Explain why it would be useful for a Web site to include a customized stylesheet for its CAP alerts.

CAP 103 - XML for CAP Implementors 53

What have you learned?

1. Explain what is XML and where the definitions of CAP elements are found.
2. Describe how elements contain other elements, such as the CAP *headline* sub-element of the *info* element.
3. Explain the basics of creating and validating a CAP alert according to XML syntax and a specific version schema.
4. Distinguish among the XML for a CAP alert and the XML for an RSS news feed pointing to CAP alerts.
5. Explain why it would be useful for a Web site to include a customized stylesheet for its CAP alerts.



Reference Links

- XML Tutorial by w3schools.com
- CAP Information Site by WMO Public Weather Services (PWS)
- WMO PWS CAP Jump Start Offer
- OASIS Emergency Management Technical Committee
- International Register of Alerting Authorities
- Google Public Alerts
- Intro to CAP (10-minute video) YouTube FTP

CAP 103 - XML for CAP Implementors 54

Here are some key reference links concerning CAP.

This concludes my presentation. Thank you for your attention.